A MULTIPLE MC68000 MICROCOMPUTER SYSTEM

by

CHIEN-CHI SYU, B.S.

A THESIS

IN

COMPUTER SCIENCE

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

MASTER OF SCIENCE

Approved

August, 1986

ABSTRACT

In this thesis, the design, construction and evaluation of a multiple processor microcomputer system are presented. The design logic of the system is discussed through a review of multiple processor systems which exist today. Their problems are then surveyed. A flexible system consisting of modules of Single Board Computers (SBCs) and Dual Port Memories (DPMs) is developed to try to solve the problems existing in the current systems. Interprocessor communication hardware is constructed to perform the interfacing and timing requirements between control units and the dual port memories. Benchmark programs are designed and coded in the MC68000 Assembly Language to evaluate the system.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The speed of system components will not continue to increase as fast as it has in the past, because, according to the laws of physics, the speed at which a digital computer can transfer units of information is approaching the upper bound of the electronic conducting capability of semiconductors. However, system performance requirements continue to grow. In order to increase system performance, system expansion capability should be provided to allow additional processing power to be added without the need for the total redesign of existing systems. We must investigate changes to classical system architecture so that system upgrading in the near future is possible [ENSL74].

In many control applications of computer systems, reliability and availability are very important features in that a failure can result in the damage of equipment and the material processed, and pose a threat to the safety of people working in the environment. Therefore, computer systems which have high performance, expansion capability, and reliability, are needed [JOHN84, BALP81, TEDD84], LEIB85].

A multiple processor microcomputer architecture is a major candidate to provide the increased performance,

reliability, and availability required. In this chapter, the advantages of multiple processor systems which match the demands of today's computing environment are described. Several different designs and implementations of multiple processor systems developed in the near past, and their problems, are then discussed.

1.1 Advantages of Multiple Microprocessor Systems

There are many advantages of using multiple microprocessor systems. The following lists some of these.

1. High performance/cost ratio. The performance of a multiple processor system is higher than a single processor system using the same processing unit, because of the additional availability of processing power from the additional processors. Recently, some new requirements and complex applications, which have an explicit parallel nature of computation, need additional processing power [GRIN85]. In the past, an additional processing unit was either too expensive or was not powerful enough to meet these requirements. With advances in microprocessor technology, the cost of processing power can now be ignored when compared with the cost of other system parts. Therefore, it is very attractive to use

additional processors in applications which require extensive computation, such as data processing in weather prediction systems and target identification in radar systems. In these situations, no additional expensive peripheral devices are needed. The performance/cost ratio increases when a multiple processor system is used.

2. High throughput. System throughput increases as the number of processors in the system increases. When a high throughput system is required, it is easier to build a properly configured multiple processor system from off-the-shelf processors than it is to redesign and construct a new, expensive processor with higher throughput.

3. High reliability. Some specific environments and situations, such as applications in hospitals, space shuttles, nuclear power plants and military equipment, demand highly reliable processing power to meet their requirements. Most multiple microprocessor structures are redundant in nature, and reliability is improved by their duplicated hardware elements and by software tasks. A failure in one processor is usually not fatal; the system can keep on operating with some degree of degradation. Thus, multiple processor systems are more reliable than uniprocessor systems.

4. High availability. Most multiple microprocessor systems have high availability because of their longer mean time before failure and shorter time to repair compared to single processor systems. These features are due to redundancy and the fact that in most multiple microprocessor systems, the processing units are similar or even identical to each other, which means a faulty unit can be easily replaced by a spare. In single processor systems, the failure of a single IC chip could result in the breakdown of the whole system. Besides, it is easier to locate a faulty module in a multiple processor system than to find a faulty component in a single processor system. It is quite different in a single processor system in that the whole system must be turned off if a faulty processor is to be replaced by a new one.
5. Concurrent system development and utilization. A multiple processor system has a greater potential for use even when it is only partially built. Even more, the development, implementation, and installation phases can be overlapped. Once a functionally independent unit is developed, it can be implemented, installed and used.
6. High expanding ability. Subsystems can be added to an existing system with no increase in memory contention and the least interruption of operation.

Of course, there are disadvantages of multiple processor systems. We should consider other costs associated with multiple processor systems, such as complicated software, component boards and connectors, which will reduce the performance/cost ratio. The difficulty to recode a sequential program into a parallel one is another problem. Also, when we incrementally increase the number of processors in a multiple processor system, throughput increases at first; then it becomes saturated. This saturation (or reduction in throughput) occurs because an increase in the number of processors results in an increase in the amount of contention of the shared resources and overhead of information exchange due to interprocessor communication.

However, multiple microprocessor systems definitely will have many applications in the future [HORD85, HALL80].

1.2 Investigation of Current Systems

Motorola, Incorporated developed the MC68000 microprocessor in 1979. D. Gosman, L. O. Hertzberger, and G. Kieft implemented a multiple microprocessor system with several MC68000s in 1982. This is the only such system, which uses this particular microprocessor, that has been reported publicly [GOSM82]. The main features of this system are as follows:

1. The system is a hierarchical tree hardware structure, which consists of an on-line filter, to select useful data and discard useless data, and a second stage trigger to start working when data collected by the filter reaches some critical level.
2. It is based on an MC68000 from Motorola, Incorporated.
3. System software supports data acquisition and operating system modes. The data acquisition mode provides the user with facilities to run time-critical programs. The operating system mode includes the kernel, the minimum operating system and the extended operating system to supervise the functions of different levels in the whole system.
4. System interprocessor communication is handled by dual port memories between the slave processing cells and the supervisor through "broadcasting write" and "wired-or read" operations. One of the dual ports is realized as a back-panel connector to interface with the internal system bus, while the other is realized as a front-panel connector to the external bus. Each port has its own address decoding PROM so that the same memory has different address spaces in the memory maps of the associated processing cells. No message center is provided and no data exchange is permitted between slaves.

5. Each processing cell consists of a CPU module and other system components such as RAM, EPROM and an I/O interface to the host, data-taking computer.

This mid-size system is used to perform calculations encountered in nuclear physics. The slave processing cells are arranged in a pipelined architecture with no direct interaction between them. It is a special-purpose multiprocessor system restricted to one application. Expansion of this system is possible, but the application is still limited.

For comparison, the following refers to two recent, representative works in small scale multi-microprocessor systems. Paul M. Russo [RUSS77] designed a multi-microcomputer system using the COSMAC microprocessor in 1977. The COSMAC microprocessor is implemented on a single 40-pin C2L MOS/LSI chip. The main features of the COSMAC are as follows:

1. 8-bit multiplexed address bus (16-bit address register).
2. 8-bit data bus (8-bit data register).
3. One-byte instruction format.
4. No Multiply and Divide instructions.

As far as extensive memory access is concerned, this system is inherently slow because of its 8-bit structure. The number of instructions is quite small because of its

one-byte instruction format. The multiplexed address bus complicates the hardware design. Not using built-in instructions for Multiply and Divide makes scientific applications slow and inconvenient. The interface philosophy of this multi-microprocessor is as follows:

1. A master-slave organization is used.
2. No common memory is provided.
3. Each CPU has sufficient RAM to handle its dedicated workload.
4. The master CPU has sufficient RAM to buffer all information to be exchanged between it and all the slave processors.
5. Interprocessor communication can be either via programmed-mode I/O or via Direct Memory Access (DMA) for block transfer.
6. The master can interrupt any slave, but no slave can interrupt the master processor.
7. Information exchange between slave CPUs can only occur via the master.
8. The master's DMA channel is time-multiplexed between the various slave CPUs under its control. The master-slave interprocessor communication scheme makes interaction between slave CPUs slow, because their communication can only occur via the master. There is a specific time

delay in data transfer, especially when the number of slaves increases, since time-multiplexed data transfer is used.

Three engineers in the Laboratory of Electronics and Microprocessors at Rockefeller University, New York, New York, Gordon Silverman, Avram Stundel and John Lehman, designed a multiprocessor-based instrument in 1982, using a set of microprocessor-controlled building blocks with a shared bus and common memory to handle their interaction [SILV82]. The main features are:

1. Each block is a Single Board Computer (SBC) built from an Intel 8085A 8-bit CPU and other system components.
2. One of the processors was configured as the executive.
3. The executive supervised a dual-port RAM which served as the common memory for global data and message exchange.
4. Some processors are configured as consumers (consume data) and others as producers (produce data). Only the producers can write to the common memory, so the function of this system is limited to very specific applications.
5. A mailbox is used as the interprocessor communication scheme and interrupts are adopted to start communicating messages.
6. The system bus supplies the address, data and control lines that allow the various components to interact through an external arbitration unit.

7. It is a master-slave structure in that a bus-master takes control of the bus and the other devices are prevented from gaining access to it. Message exchange is limited and communication speed slows down, because task cooperation between processors in the system must go through the executive.

1.3 Problems in Current Systems

The main problems concerned with the current multiple microprocessor systems are the following.

1. The system organization of multiple microprocessor systems which exist today are limited to specific applications. When the application changes, they become useless. A new system must be built, resulting in a waste of resources and implying additional financial investment.
2. They are mostly composed of 8-bit microprocessors which are slow, when extensive memory accesses are required, and hard to program. For scientific applications, which need extensive arithmetic computations, microprocessors with 16 bits or more of data bus and high computation capability must be used [KART82, TOUN81].
3. They usually use common memory to manage data flow between processors, which will result in bus and memory

access conflicts. Therefore, considerable software overhead is required to obtain system synchronization.

1.4 Preview

Many bus and memory conflicts are the major problems which exist in many current multiple microprocessor systems, such as process control systems which use a common memory for message exchange. In the following chapters, we explain how a system structure, built around MC68000s as the control units, using dual port memories which have no bus conflict and minimal memory conflict as the data exchange media, and having a versatile configuration ability, will help to solve these problems. A general overview of multiple processor systems, their advantages, history and problems, are contained in this chapter. Chapter 2 describes different configurations and expansion capability of the system developed for this research. The remainder of this thesis is devoted to details of this system. The functions of the control unit and the interprocessor communication medium are presented in Chapter 3. In Chapter 4, the hardware details of the interprocessor communication unit (the dual port memory) are addressed. In Chapter 5, we briefly describe an actual process control example and test programs which simulate a process control system. Chapter 6 summarizes this thesis

project and evaluates the system designed, using tables and figures obtained from execution of the test programs. Suggestions for future extensions are discussed at the end of the concluding chapter.

CHAPTER II

SYSTEM CONFIGURATIONS

In this chapter, system structures, which consist of Single Board Computers (SBCs) and Dual Port Memories (DPMs) and provide true concurrency, reliability, ease of design and reconfiguring ability, are discussed. These system structures are not restricted to any particular application. The modules in this structure can be easily rearranged to form new configurations for new applications.

2.1 System Reconfiguration

Depending on situations encountered, there are two possible ways of reconfiguring systems containing SBCs and DPMs: dynamic and static. Dynamic reconfiguration is done during run time, using software stored in the local memory of each SBC. This software reflects the number of SBC and DPM modules in the system and the connecting paths between these modules. On the other hand, static reconfiguration is performed by physically connecting or disconnecting some SBCs and DPMs in the system before the system is turned on.

In this section, we describe possible configurations of three SBCs. Configurations using more than three SBCs and DPMs are discussed in the next section.

1. Delta Configuration. This consists of three SBCs to form a master-master type multiple processor system. Each SBC works independently and can communicate with two other SBCs through the dual port memories connecting them. The Delta configuration is shown in Figure 1.
2. Pipeline Configuration. In this configuration, data enters one end of the structure and is processed by each of the processors in the system. Finally, data exits the other end (See Figure 2).

2.2 Expansion Capability

If we expand the system of three SBCs and three DPMs by adding one more SBC and three more DPMs, we will have a fully connected system with four processing nodes and six branches, forming a Pyramid structure. This structure can be reconfigured into other structures by ignoring some nodes and/or branches in the system. The following explains the possible system configurations:

1. Pyramid Configuration. This is a symmetric system with high flexibility and reliability. It is a reliable interconnection scheme because it provides alternate paths when a direct path between two unit fails. With its high reconfiguration ability, this system can be used as a unit to form a large scale system which may include

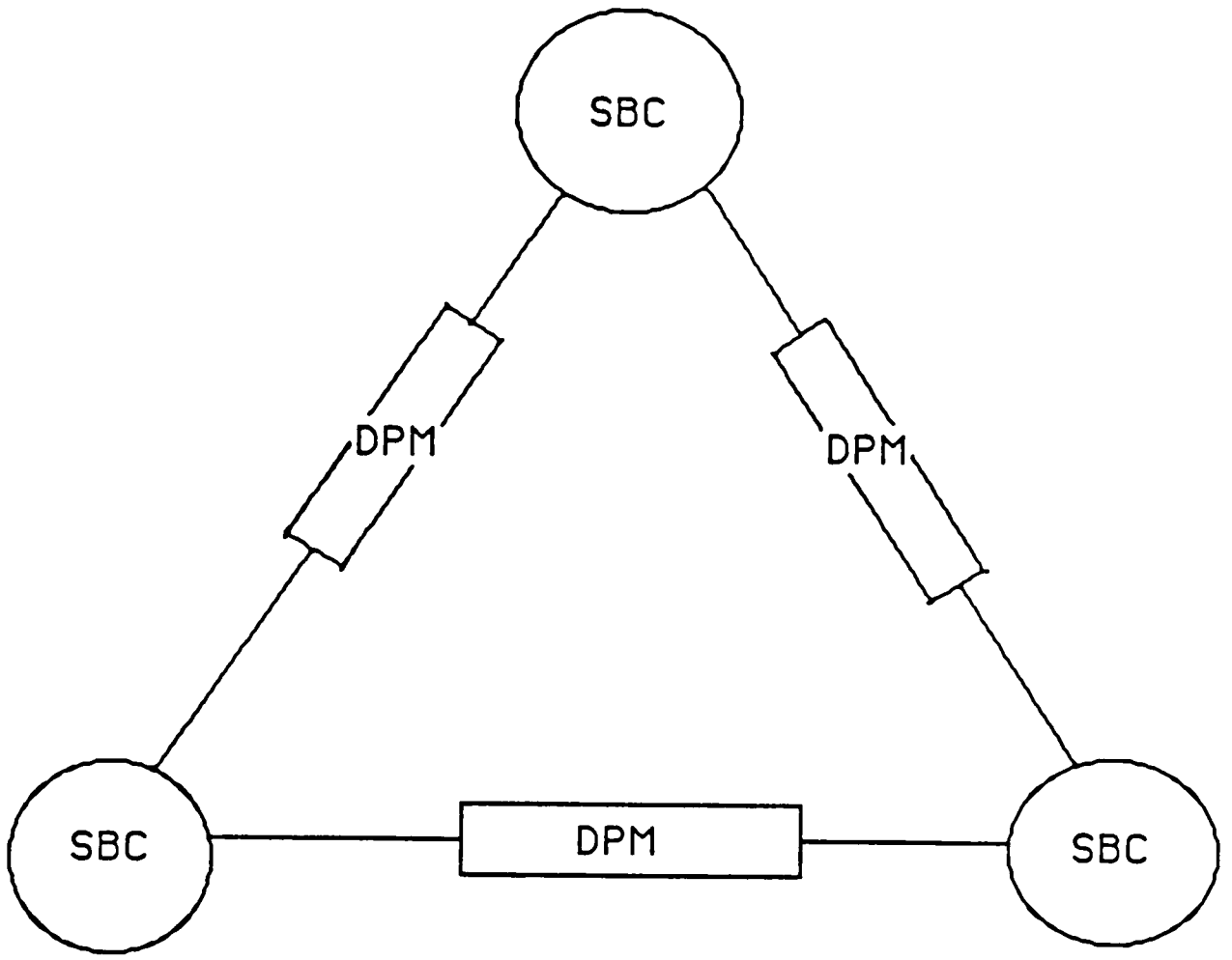

Figure 1: The Basic Delta Configuration

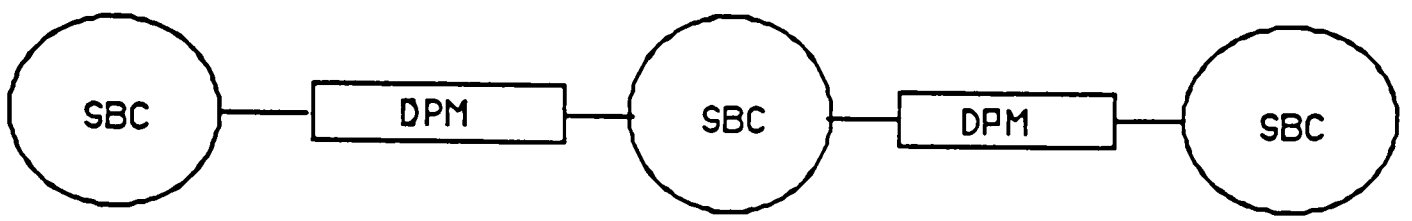

Figure 2: The Pipeline Configuration

hundreds of SBCs. The Pyramid configuration and the Delta configuration given previously are symmetric graph structures such that any one of the SBCs can communicate with all other SBCs in the system and any of them can be assigned as a master. Figure 3 shows the Pyramid configuration.

2. Delta Configuration: The Pyramid can be reconfigured into a Delta Structure by disconnecting at the points indicated by asterisks in Figure 4.
3. Square Configuration: The Pyramid can be reconfigured into a Square Structure by disconnecting at the points

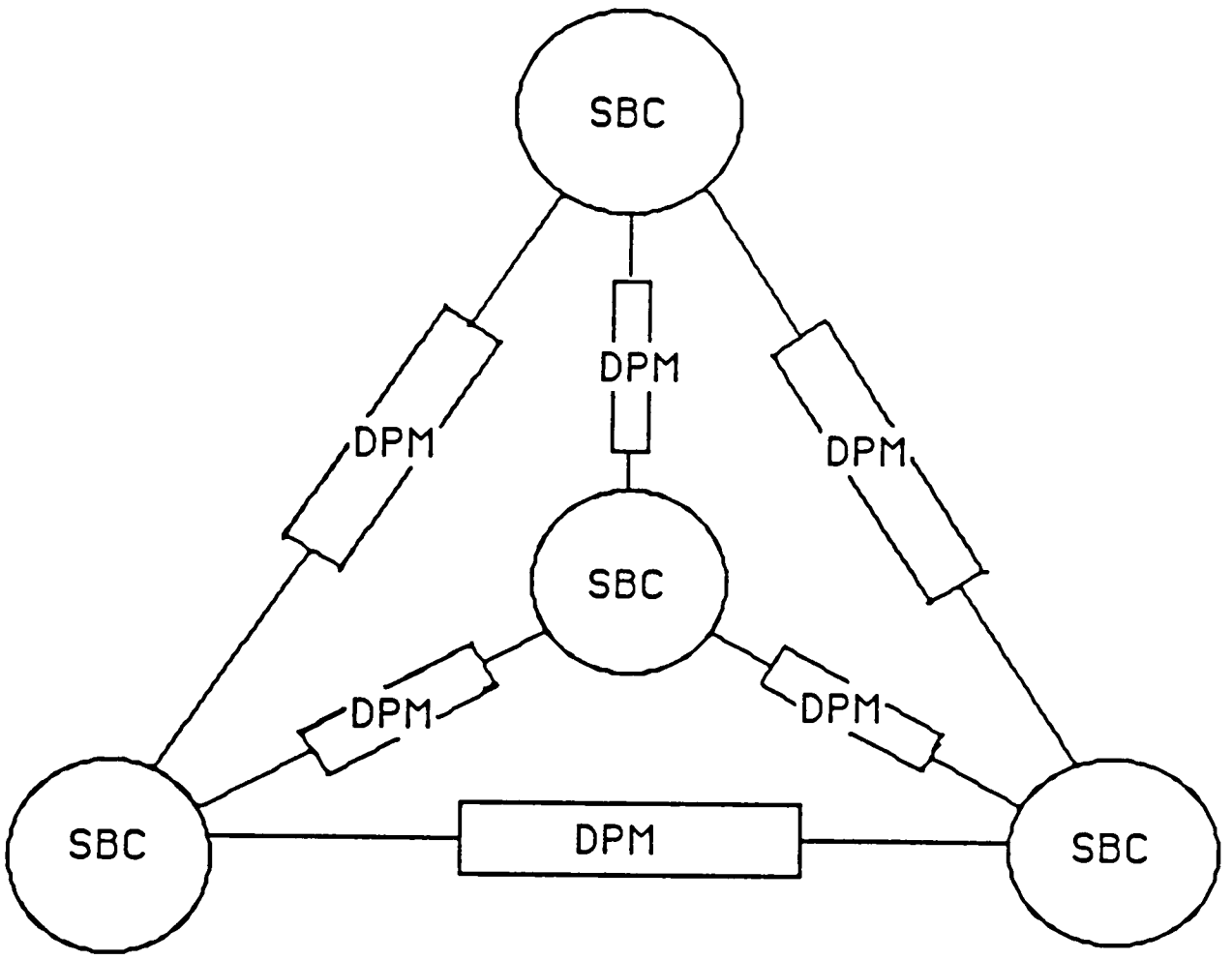

Figure 3: The Pyramid Configuration

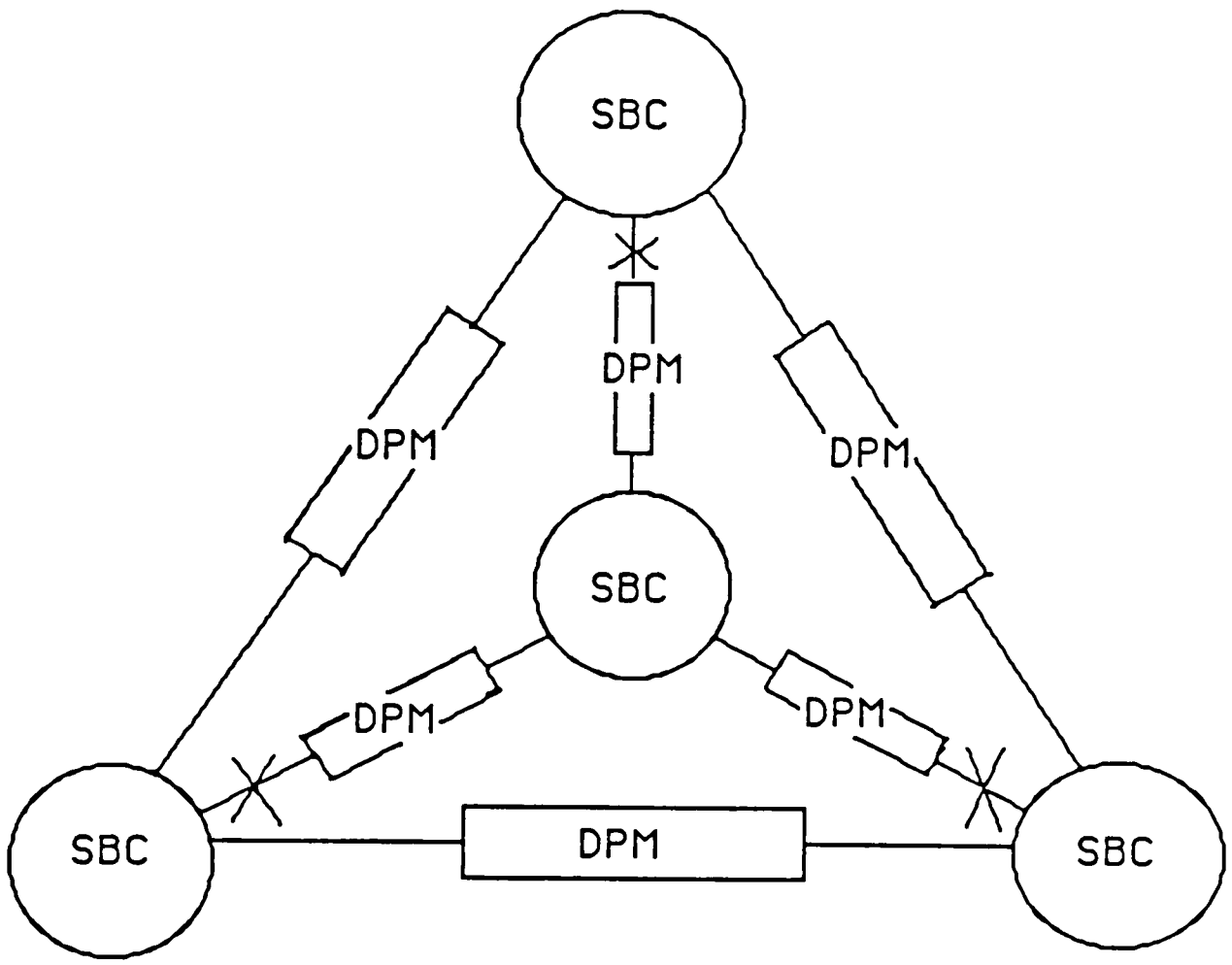

Figure 4: The Delta Configuration

indicated by asterisks in Figure 5. Note: The Delta and the Square Configurations are examples of Ring Structures.

4. Star Configuration. The Pyramid structure degenerates to a Star structure if the three outer dual port memories are actually taken away or are ignored by the software in real time. The Star configuration is a master-slave system and is useful in redundant processing with the central SBC acting as a checking processor [ENSL74, BOZY82, VIVE82]. The Star Structure, reconfigured from the Pyramid Structure by disconnecting at the points indicated by asterisks, is shown in Figure 6.

5. Pipeline Configuration. We can also reconfigure the Pyramid into a Pipeline of three (or four) SBCs and two (or three) DPMs by making appropriate disconnections in the Pyramid.

The expansion of this structure to include N SBCs with a maximum of 1/2 N(N-1) DPMs is possible, except that the value of N is limited by the memory space addressable by the address bus of the control unit. For the system developed for this research, and described in the remainder of this thesis, we use a Motorola MEX68KECB as our single board computer. On the MEX68KECB, there are seven connecting points from the address decoder which can be used to enable

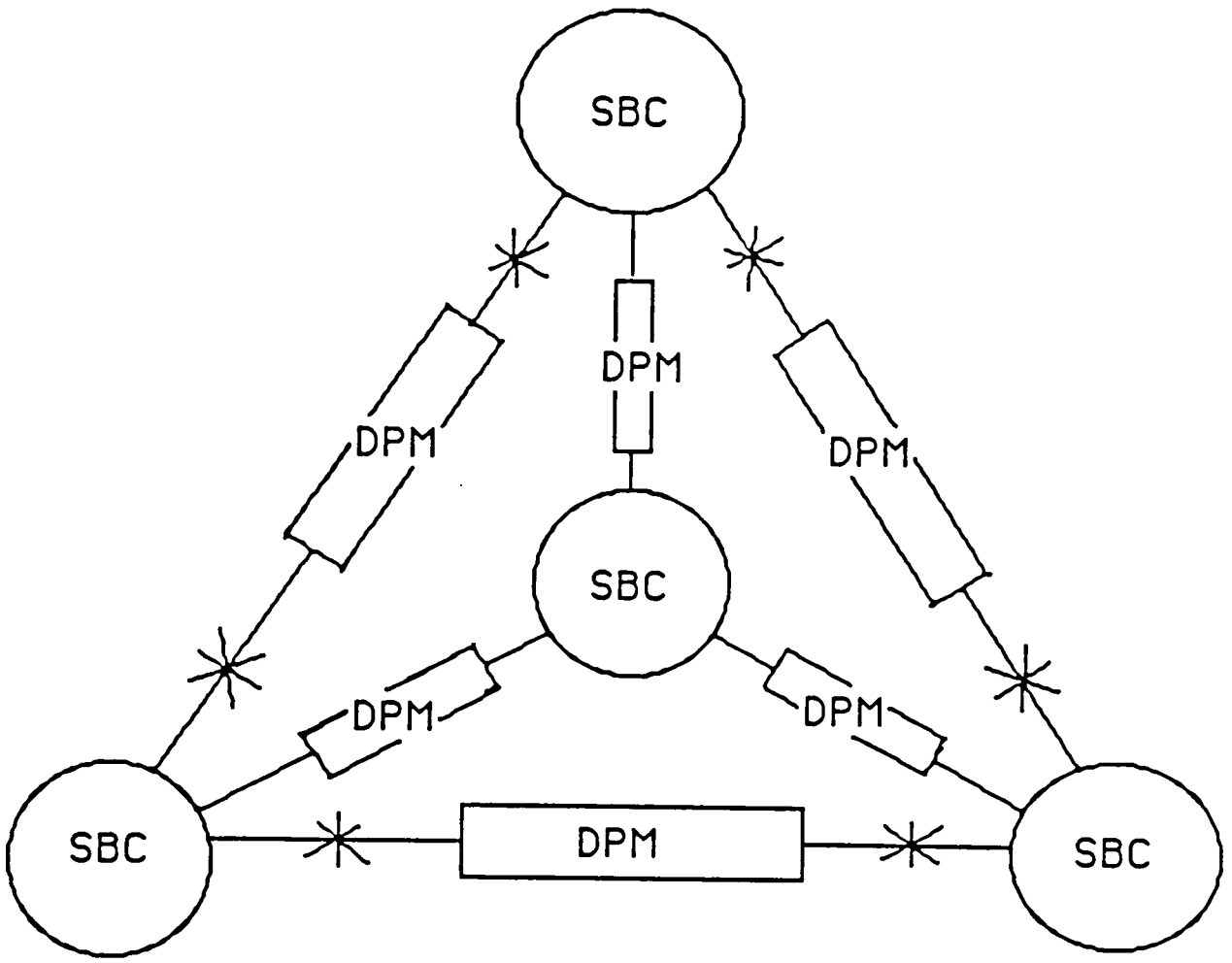

Figure 5: The Star Configuration

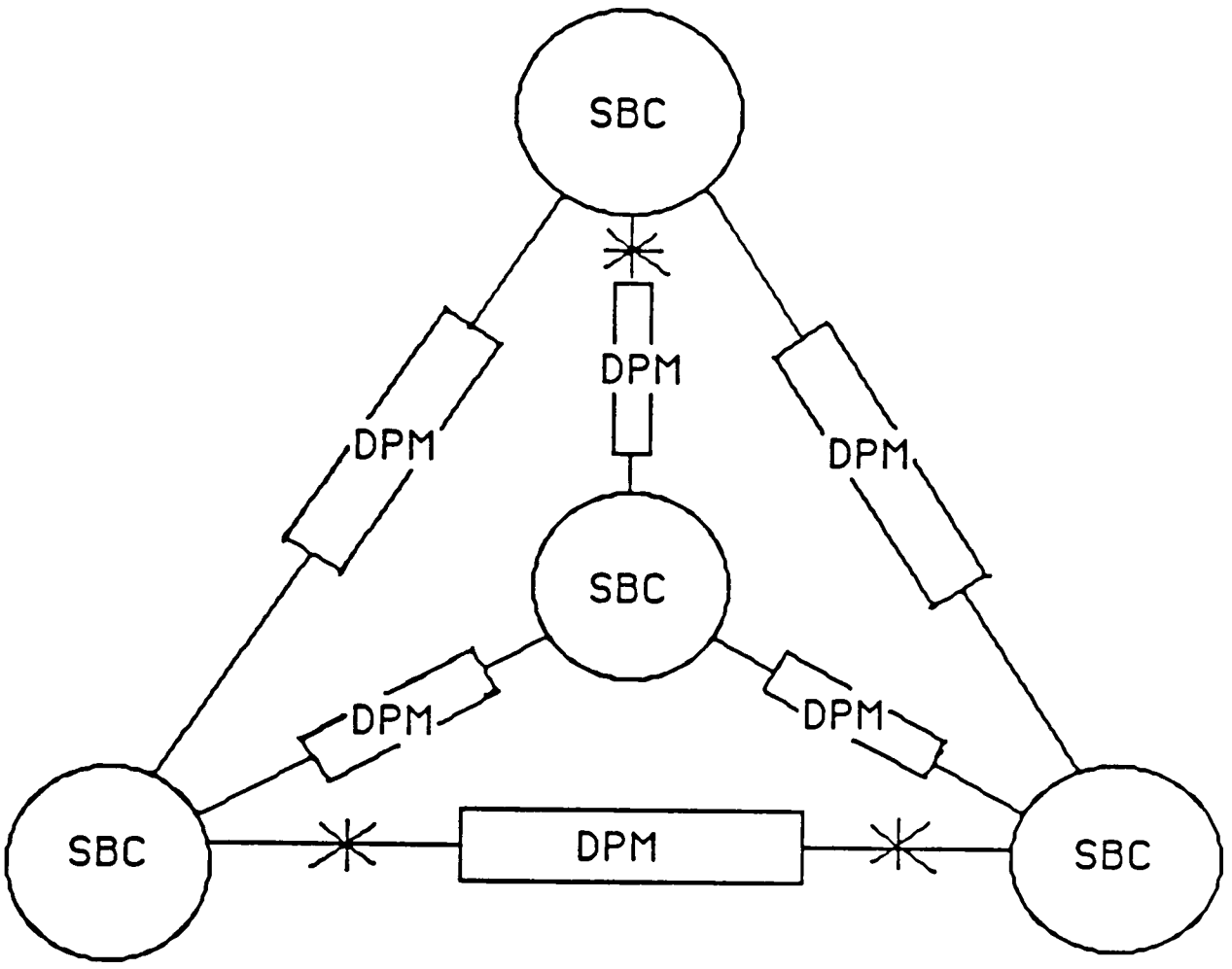

Figure 6: The Square Configuration

user expanded memory units. Thus, eight SBCs are allowed in our structure and a maximum of twenty eight DPMs could be included to perform message transfer between any two processors.

2.3 The System Configuration Developed for this Research

The structure designed in this research does not have the complexity a general purpose multiple processor system with common memory usually has; for example, our system includes neither control nor address buffers for each processor, and no external arbitration logic for these buffers. It does provide the capability of being configured into different special purpose multiple processor systems. A system for a particular application can easily be realized from this structure by plugging in the proper number of SBCs and DPMs with appropriate interconnections [STAR80, MOTO82].

Three MC68000-based SBCs, the MEX68KECBs from Motorola, Incorporated and three TMS9650-based DPMs from Texas Instruments, Incorporated were included in this implementation. These modules (SBCs and DPMs) were connected into a delta (or triangle) type system when it was tested and evaluated. The topology of this system is shown and discussed in the next chapter. However, these modules could be connected in

different ways, or modules might be added to (or removed from) this system, when application requirements change.

In applications which require frequent and quick reconfigurations, the system developed is an effective and economical one because of its modular structure.

2.4 Preview

In the next chapter, the MC68000 control unit and the TMS9650 dual port memory unit are discussed first. The Test And Set instruction which is used in this multiple processor system implementation is then introduced.

CHAPTER III

SYSTEM COMPOSITION

The functions of the control unit (the MC68000) and the configuration of the dual port memory (the TMS9650) used in the system design are covered briefly in the beginning of this chapter. A diagram of the topology of the system and an explanation of the TAS instruction, used for the system synchronization in the multiple processing environment of this implementation, also appear in this chapter.

3.1 The Control Unit

The MC68000 [MOT083A] is used as the control unit because it provides a longer word, more addressing modes, an extensive and powerful instruction set, higher operating speed, lower power consumption and more software support for multiple processor system applications than most other popular microprocessors used today. The Motorola Educational Computer Board, MEX68KECB, is a uniprocessor microcomputer system (an SBC) based on the MC68000 [MOT082]. Other system components included in this SBC are on-board memory, three I/O interface chips, two MC6850 ACIAs (Asynchronous Communication Interface Adapters) for serial interface with a terminal and host computer, and one MC68230 Parallel

Interface/Timer for a parallel interface with a printer and/or a cassette tape recorder. The on board memory includes two MCM68A364 ROMs for the system firmware, the TUTOR monitor, the RESET routine vector, sixteen MCM4116 dynamic RAMs to store other exception handling routine vectors, system programs, user programs and data [MOTO81,MOTO82].

In this research, three MEX68KECBs and three TMS9650 DPMs [TEXA848] are assembled to implement the delta configuration of the multiple processor structure described in Chapter 2. Each SBC has an ACT-5A terminal for its system console. The system topology is shown in Figure 7. Each of the three MC68000s simultaneously executes TUTOR in its local ROM when power is turned on. Test programs are downloaded to the local memory of each MEX68KECB from the Hewlett Packard HP64000 Logic Development System [HEWL80] and the proper starting addresses of the programs are assigned to each program counter. TUTOR runs the test program when the GO command is entered through the terminal. For real world applications, a more sophisticated operating system is needed to provide additional functions.

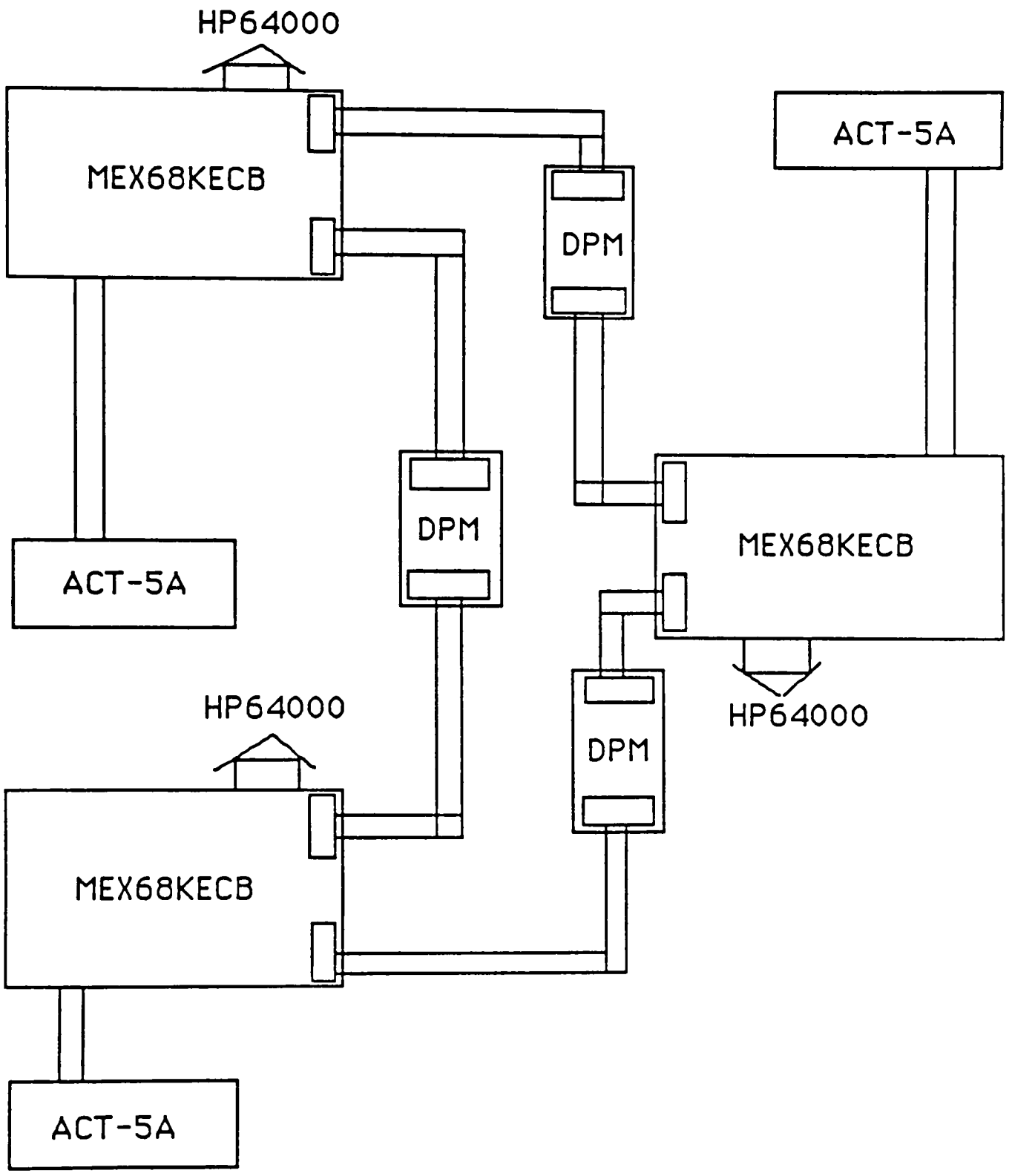

Figure 7: The System Topology

3.2 The Dual Port Memory

A dual port memory is used as an interprocessor communication medium between each pair of processors in this multiple microprocessor system. This type of structure introduces no bus conflict because the dual port memory prevents the system bus of an individual processor from interacting with any other. Each processor may gain control of the bus whenever a bus cycle is needed, and access the common dual port RAM in the same manner as it accesses its local RAM. Memory conflict is reduced to a minimum, because only two processors contend for each DPM, and contention is usually resolved in the logic of the dual port memory. The TMS9650 IC chip consists of one 256-byte static RAM, eight 8-bit registers, control pins, and arbitration logic. These features are briefly described in the following [TEXA848].

1. The Registers. In the following register descriptions, "local" refers to the port connected to the SBC which is accessing the DPM and "remote" refers to the port connected to the SBC on the other side of the DPM.

   (a) Data Register. This acts as a data buffer between the data bus and static RAM. The data to be accessed is put in the Data Register before it is transferred to the RAM or to the MC68000 registers. The Data Register address in the memory map of the MC68000 is

0y0009 (where y is either 2 or 4 depending on which DPM is to be addressed). The addresses of the TMS9650 registers are listed in Table 1.

TABLE 1

The Addresses of the TMS9650 Registers in the MC68000 Memory Map

Address Lines (hexadecimal)		Register Select Lines				
A23---A0		A3	A2	A1	A0	Register Selected
DPM4	DPM2	S2	S1	S0		
040001	020001	0	0	0	1	Data/Inc
040003	020003	0	0	1	1	Control
040005	020005	0	1	0	1	Message In
040007	020007	0	1	1	1	Status
040009	020009	1	0	0	1	Data
04000B	02000B	1	0	1	1	Local Address
04000D	02000D	1	1	0	1	Message Out
04000F	02000F	1	1	1	1	Remote Address

(b) Data/Inc Register. This has the same function as the Data Register, except that, when it is accessed, the content of the corresponding Local Address Pointer is

incremented by 1 to point to the next location in the RAM. The Data/Inc Register address is 0y0001.

(c) Message Out Register. This can be used to transfer one byte between the two S8Cs. Its address is 0y000D.

(d) Message In Register. A byte transferred from the remote side of the Message Out register appears on the local side of the Message In Register. The address is 0y0005.

(e) Local Address Pointer. This specifies the RAM location the MC68000 is going to access. The address is 0y000B.

(f) Remote Address Pointer. The Local Address Pointer is known as the Remote Address Pointer by the MC68000 on the other side of the DPM.

(g) Control Register. This is used to enable or disable the bits in the Status Register and has address 0y0003.

(h) Status Register. This register indicates the interrupt status of the TMS9650 and issues an interrupt signal to the MC68000 through the output pin INT on the TMS9650 chip. 0y0007 is the address.

2. The static RAM. The RAM is accessed by both interfaces via Data Registers. It can only be accessed by one

microprocessor at a time. The two outputs of the arbitration latch, ACTA and ACTB, control access to the RAM. If we need to write a message with value, VAL, into a location with value, LOC, in the RAM, we first write the value, LOC, into the Local Address Pointer, then we write the value VAL into the Data Register. On the other hand, the read operation is performed by writing a proper address into the Local Address Register and then reading from the Data Register.

3. The arbitration logic. The arbitration latch is responsible for granting RAM access to one port or the other to solve possible contentions between the MC68000s. When a contention is resolved, and access is given to one of the MC68000s, the READY signal is driven high to the corresponding side of that MC68000 so that the bus cycle may proceed.

4. The control lines.

   (a) RESET*s from MC68000s are connected to M1 and M2, which are the Mode pins on each side of a TMS9650, to select a proper operation mode of the TMS9650 DPM.

   (b) AS* (Address Strobe Low) from an MC68000 is connected to LOCKIN* through the interface board.

   (c) Es (Enables) from the address decoder, LDS* (Lower Data Strobe Low), and UDS* (Upper Data Strobe Low),

are connected to CSs (Chip Selects) of the TMS9650 after passing through the interface board.

(d) The output signal line READY from the TMS9650 enters the interface circuit to generate the DTACK* (Data Transfer Acknowledge Low) signal.

(e) R/W* (Read or Write Low) from an MC68000 is used to generate OE* (Output Enable Low), and WE* (Write Enable Low) to the TMS9650.

3.3 The Test And Set Instruction

The MC68000 assembly language includes the TAS instruction, Test And Set, which uses a read-modify-write cycle to provide meaningful communication between processors in a multiple processor system [HARM85]. An MC68000 executes this instruction in three steps:

1. It issues a bus cycle to read data from a specific location in the DPM into a register in the MC68000.
2. It modifies the data in the register. There is no bus activity in this step.
3. It issues a second bus cycle to write the data back to the same address in the DPM.

This operation is indivisible in that the AS* is asserted throughout the three steps to allow synchronization of multiple processors; the TAS instruction is used to lock

out a shared memory (such as the DPM) from other processors when one processor has control of it. Once an MC68000 addresses an operand using a TAS instruction, the memory in which this operand is located is not available to any other processor until the instruction is completed.

The TAS has the symbolic form:

```
        TAS     EA          where EA is the effective
                            address of the operand
```

and its definition is:

```
        if      (EA) = 0    where (EA) is "contents of
                            EA"
        then    set Z = 1
        else    set Z = 0

        if      (EA)(7) = 1 where (EA)(7) is bit 7 of
                            the content of EA
        then    set N = 1
        else    set N = 0

        set     (EA)(7) = 1
```

Z and N are the Zero and Negative bits in the CCR (Condition Code Register), respectively. The TAS instruction is a byte operation; its operand is the byte pointed to by EA [MCT083A].

The hardware implementation of this specific instruction in a processing unit provides the synchronization of message transfer, which is necessary in a multiple processor environment.

3.4 Preview

In this system, we have used the MC68000 as the control unit and the TMS9650 as the dual port memory to implement a Delta structure. The TAS instruction is employed to ensure system synchronization. The next chapter is dedicated to the hardware implementation of the interface board and system connection.

CHAPTER IV

INTERPROCESSOR COMMUNICATION HARDWARE

In this chapter, we briefly describe the design, and connection of the interface board. The design requirements for control lines, the connection of address lines and data lines are mentioned. Hardware wiring details are shown in Appendix A for reference. The MC68000 Assembly Language code, used to test the functions of the system, is shown in Appendix B.

4.1 The Design of the Interface Logic

According to the functions described in the data sheet [TEXA848], the TMS9650 chip is designed to control the interface between two processors in a multiple processor system. Although the functions of the TMS9650 are close to what is needed in the design, the signal specifications are quite different from those of the MC68000. Therefore, we designed and built a logic circuit to interface the TMS9650 to the MC68000. The following describes the related specifications of the MC68000 and the TMS9650 and the logic required to match these specifications.

1. The R/W* signal from a MC68000 is high when the MC68000 issues a bus cycle to read from memory. This signal is driven low when the MC68000 is writing into memory. The corresponding signals needed by the TMS9650 for read and write operations are from two separate lines: the OE* (Output Enable Low) and the WE* (Write Enable Low). Therefore, the R/W* is ANDed and ORed with the outputs of a D-latch to produce the OE* and the WE*.
2. The higher order byte of the data bus from the MC68000 is connected to the data bus of the TMS9650 so that the address line A0 from the MC68000 is 1 when the DPM is addressed. Three additional address lines (A1, A2, and A3 which are connected to S0, S1, and S2 on the TMS9650 chip) are needed to address each of the eight registers in the TMS9650.
3. The DTACK*, which is necessary for MC68000 asynchronous bus operation, is generated using the READY output signal from the TMS9650 with the proper delay produced by a D-latch. The logic equation is:

   DTACK* = Q1* + Q4

   where Q1* and Q4 are outputs from the Quad D-latch.

4.2 The Connection of the TMS9650 and the MC68000

On the MEX68KECB, there is an area which is designed for system expansion and signal access from the outside world. The wire-wrap area is provided to mount devices for I/O port addition and memory units for memory extension. The MC68000 bus lines and system timing signals are accessed through an auxiliary I/O pin, designated J16 on the MEX68KECB. Two 20-pin sockets are mounted in this area (See Figure 8). These sockets are connected to the TMS9650 interface board with 20-wire flat cables. The pins of the sockets are connected to a connection area designated J17, which gives access to the MC68000 data, address and control buses.

The output pins E1 and E2 from the address decoder of the MC68000 on the MEX68KECB, which provide enable signals for unused areas of the MC68000 memory map, are not included in the connection area J17. They are connected to the sockets separately. The DTACK* signals are wired to pin E7 from the sockets. Open collector AND gates are used in the TMS9650 interface board for these special signals. Power supply and ground lines, Vcc and Vss, are drawn from the interior of the MEX68KECB to the sockets because they are not contained in the J17 area. The details are shown in Figure 9.

TMS9650 Signals	MC68000 Signals	Pin Number		Pin Number	MC68000 Signals	TMS9650 Signals
M1/M2	RESET*	20		1	E1/E2	X
GND	GND	19		2	A1	S0
D0	D0	18		3	A2	S1
D1	D1	17		4	A3	S2
D2	D2	16		5	DTACK*	X
D3	D3	15		6	LDS*	X
D4	D4	14		7	UDS*	X
D5	D5	13		8	R/W*	X
D6	D6	12		9	AS*	X
D7	D7	11		10	8MHZCLK	CLKIN

X signals are connected to the interface board.

Figure 8: Socket Pin Distribution

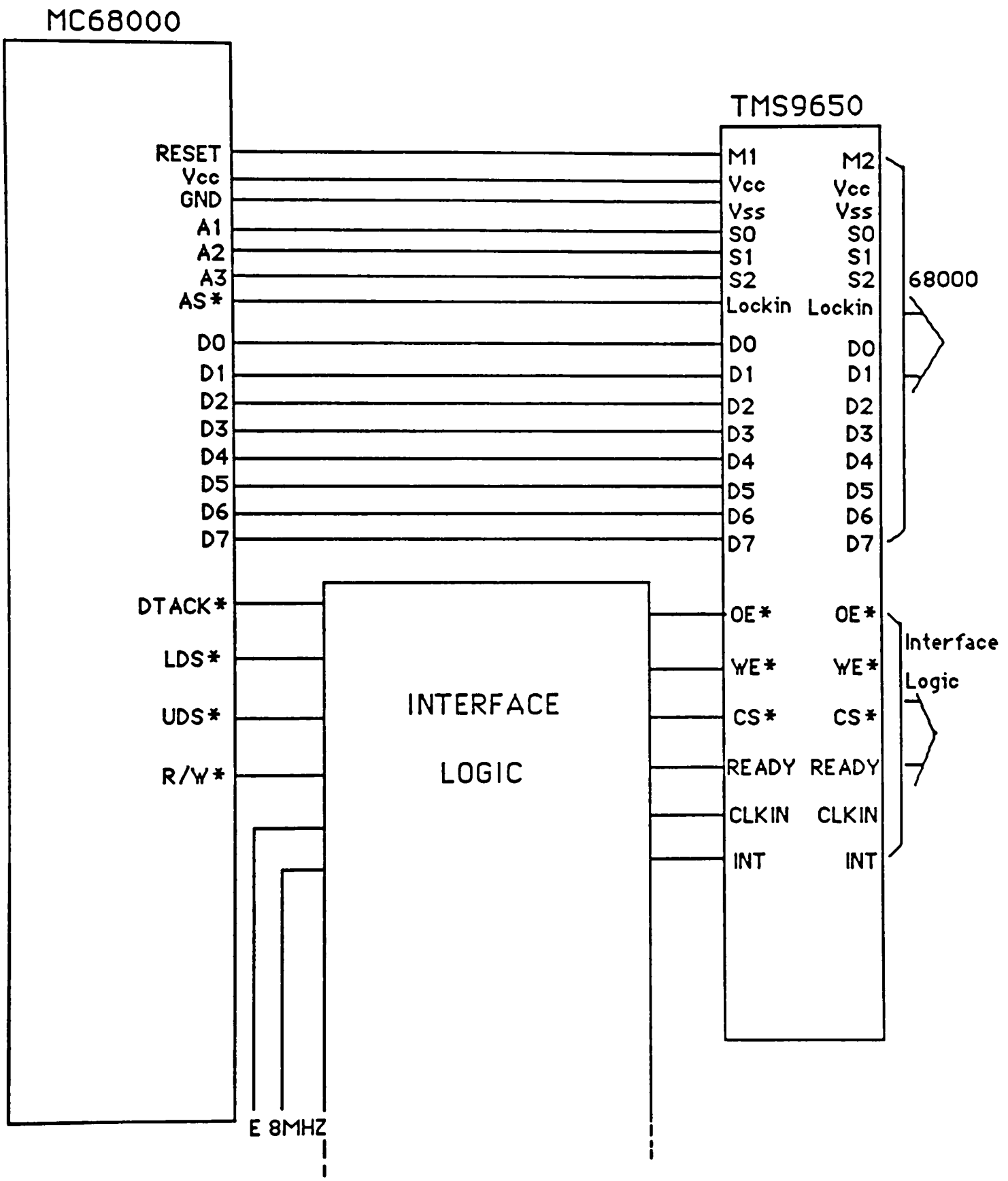

Figure 9: MC68000 and TMS9650 Connections

4.3 Preview

An algorithm for performance evaluation of the system and the measurement of the execution time of test programs, which are listed in Appendix B, are presented in the next chapter.

CHAPTER V

TESTING THE SYSTEM

In this chapter, we describe the algorithm and test procedure used to validate our system as an example of a real time process system employed in a chemical plant. At the end of the chapter we interpret the results.

5.1 The Test Procedure

Since throughput of a system is an inverse function of that system's execution time, we used elapsed job time to determine the increase, if any, in throughput of our system over that of a single SBC system. Our test procedure consisted of running nearly identical jobs, in each SBC, which executed a predefined number of loops. After each completion of the inner loop, the SBC notified its successor (the SBC to its left) in the Delta configuration. Thus, in effect, each SBC was able to maintain a count of the number of notifications sent by its predecessor in the Delta. The algorithm used in the test is shown below. Following the algorithm we discuss and display the test results.

5.2 The Test Algorithm

The sequences of the procedures which the test programs take are as follows.

0. Initialization. There are three SBCs, SBC1, SBC2 and SBC3, in the Delta test system, as shown in Figure 10. The system is started in three steps:

   (a) SBC3 is started first to run the test program. It initializes the semaphores in its DPM2 and DPM4 to zero and then enters a waiting loop, in which SBC3 keeps on polling a location in DPM4 to see if a starting message has arrived from SBC2. If yes, SBC3 goes to Procedure 1.

   (b) SBC2 is then started. It initializes the semaphore in its DPM4 to zero and then enters a waiting loop to see if a starting message has come from SBC1. If yes, it sends a starting message to a specific location in its DPM2 to start SBC3, and then goes to Procedure 1.

   (c) SBC1 is started last. It sends a starting message to a specific location in its DPM2 to start SBC2 and then goes to Procedure 1.

1. Each SBC executes a predefined number (M) of outer loops which repeat Procedure 2 to Procedure 5.

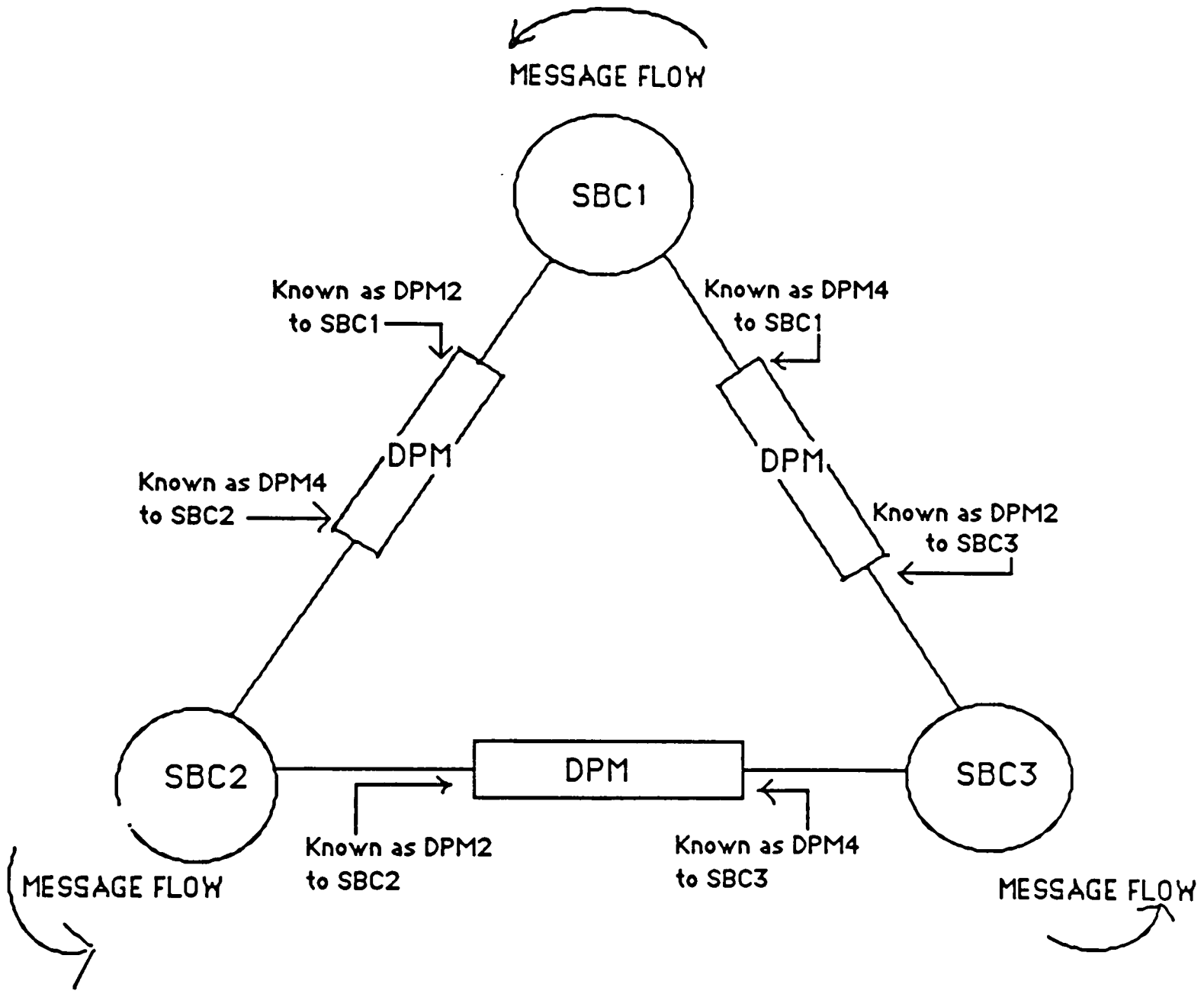

Figure 10: Message Transfer in the Developed Delta System

2. Each SBC executes a predefined number (N) of local processing loops.

3. The TAS instruction is executed before a SBC accesses a DPM to ensure two things: operation sychronization and message validation.

   (a) Sending Notification

      i Each processor checks the semaphore bit, which is bit 7 in the first byte of DPM2, to see if DPM2 is available. If yes, it then goes ahead to access DPM2. Otherwise, the processor enters a waiting loop.

      ii Each processor checks the first byte (ignoring bit 7, which is the sign bit of this byte) of DPM2 to see if it has a value of 0. If yes, the message which it previously passed has been received. The processor then writes into DPM2 (notifying the successor S8C) and sets its first byte to 1 (ignoring bit 7). If no, the processor enters a waiting loop.

   (b) Receiving Notification.

      i Each processor checks the semaphore bit, which is bit 7 in the first byte of DPM4, to see if DPM4 is available. If yes, it then goes ahead to access DPM4. Otherwise the processor enters a waiting loop.

ii Each processor checks the first byte of DPM4 to see if it has a value of 1. If yes, the message which it expected has arrived. The processor then reads from DPM4 (notification from the predecessor) and sets its first byte to value 0 (ignoring bit 7). If no, the processor enters into a waiting loop.

4. Termination. Each SBC exits from the test program and reaches a break point at which final values of all registers are displayed on its ACT-5A screen.

The flowcharts of the algorithm are shown in Figure 11 through Figure 15.

The test procedure simulates a real time system employed in a chemical plant, with several sensors hooked to each SBC to speed up the data processing rate. As shown in Figure 16, SBC1 is responsible for processing the data coming from an analog-digital convertor, which has four sensors connected to it. These sensors are used to collect process parameters, such as temperature, pressure, air density and material volume, in a chemical reaction tank. SBC1 passes process results to SBC2 through DPM1. SBC2 receives the results from DPM1, and compares these results with previous data stored in its local database. SBC2 then makes a proper decision and passes this decision to SBC3 through DPM2.

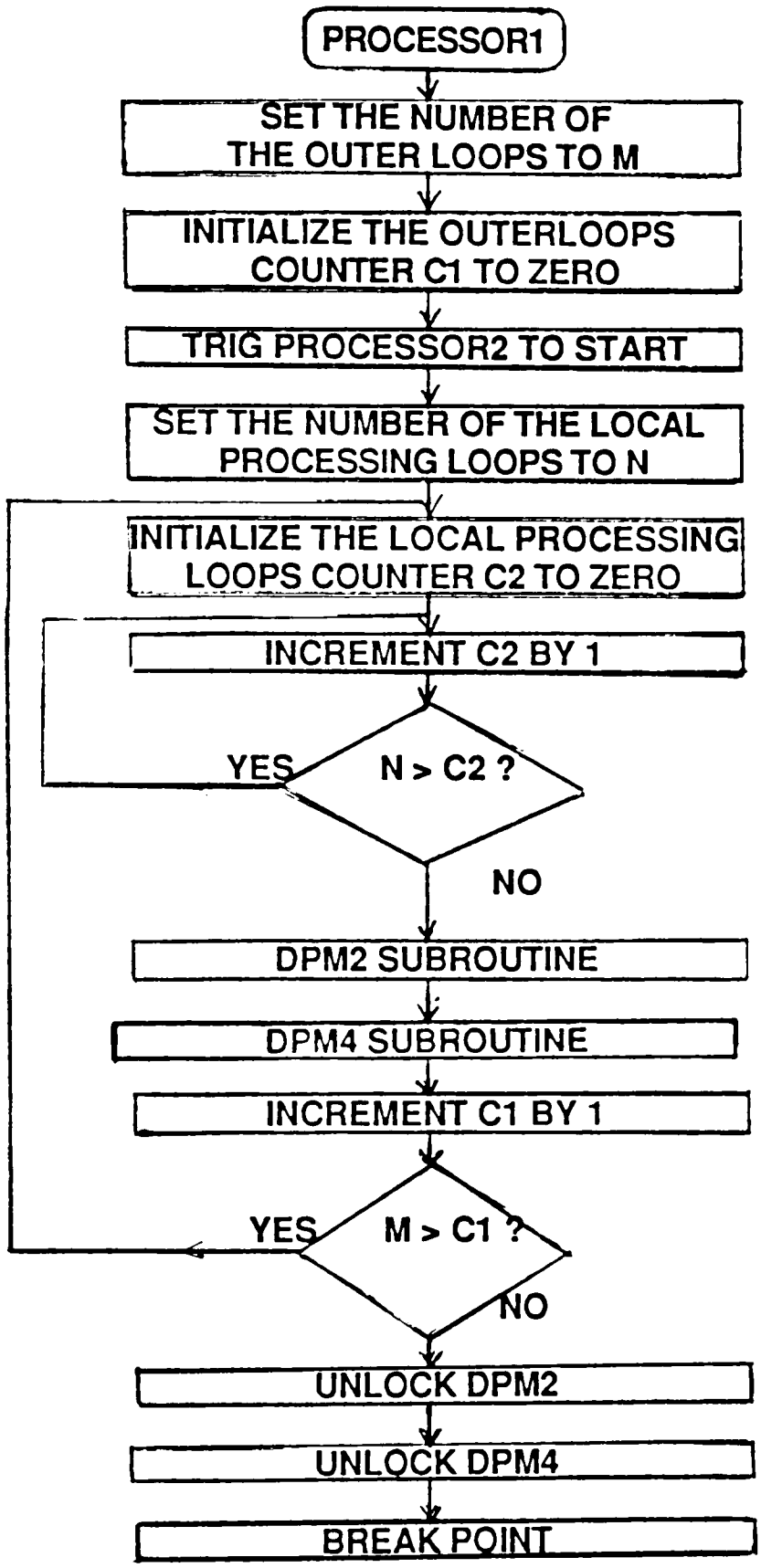

Figure 11: Flow Chart for Processor1

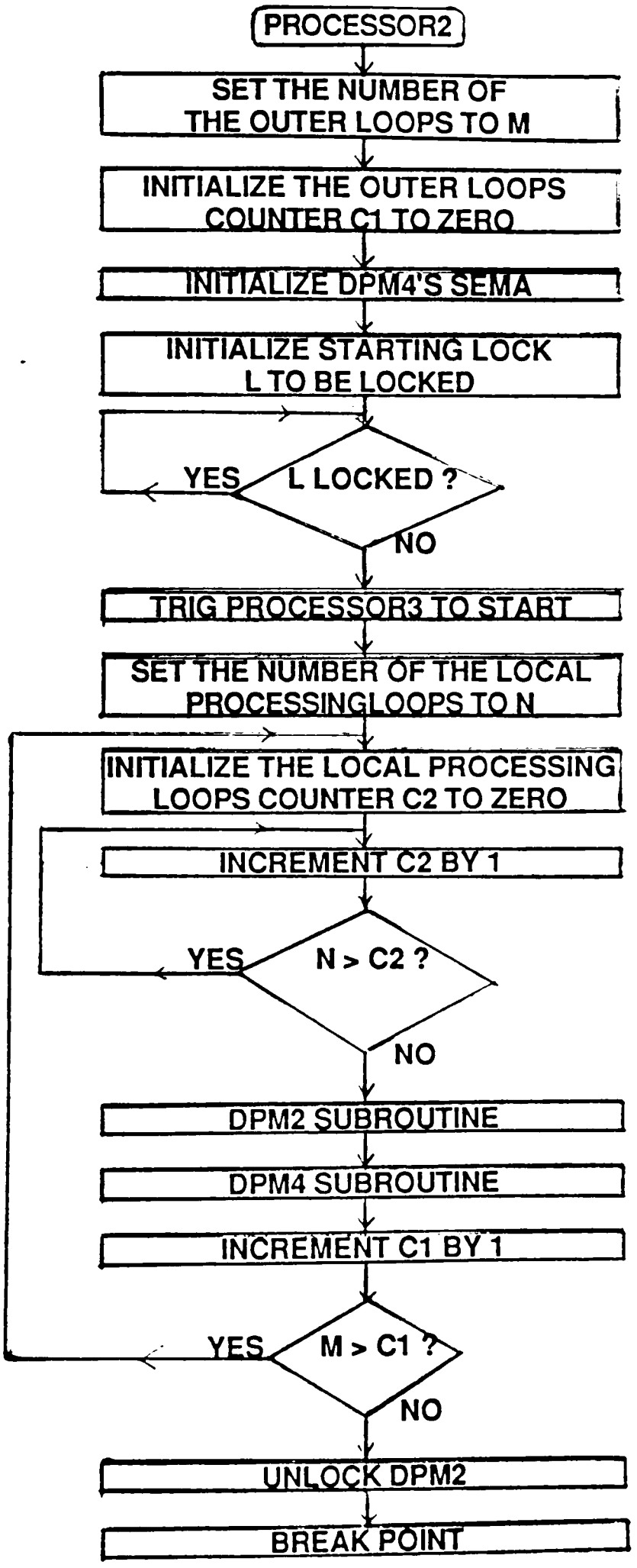

Figure 12: Flow Chart for Processor2

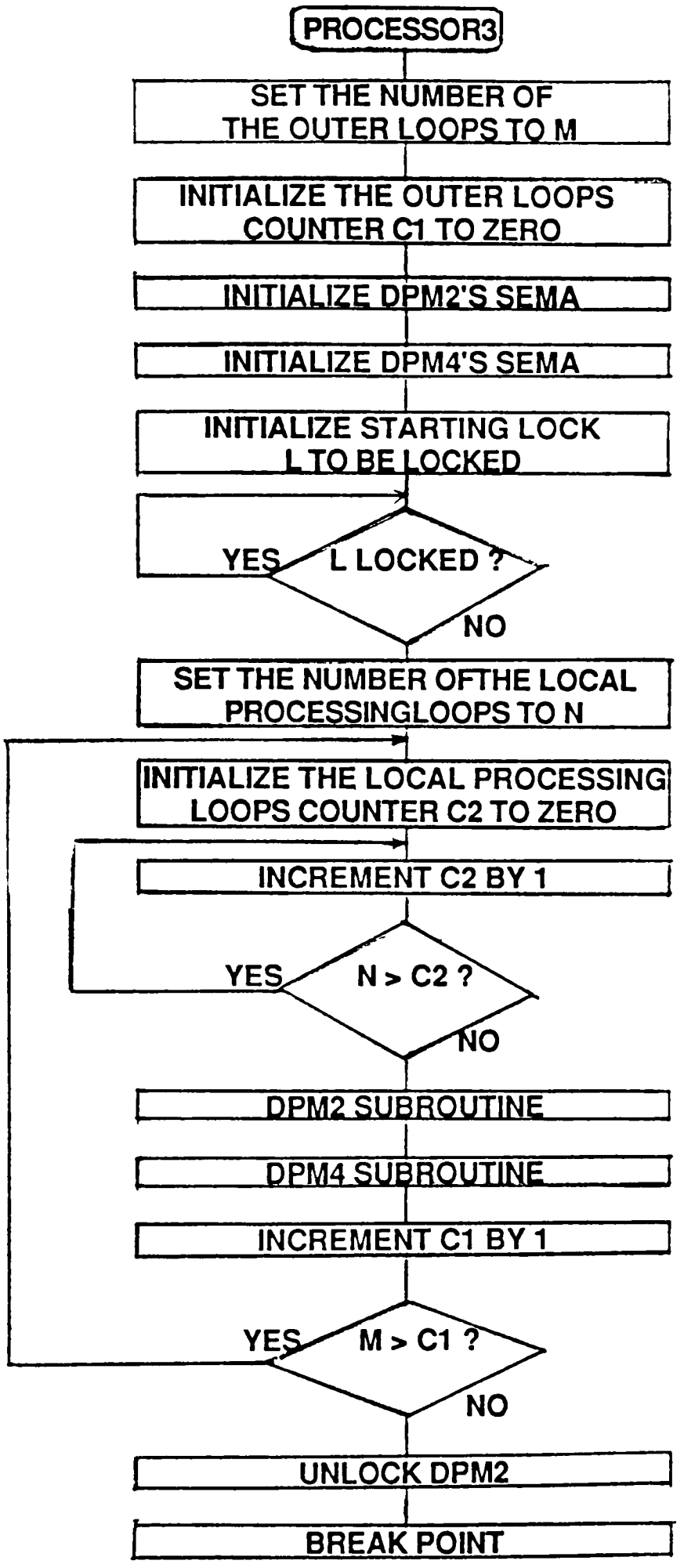

Figure 13: Flow Chart for Processor3

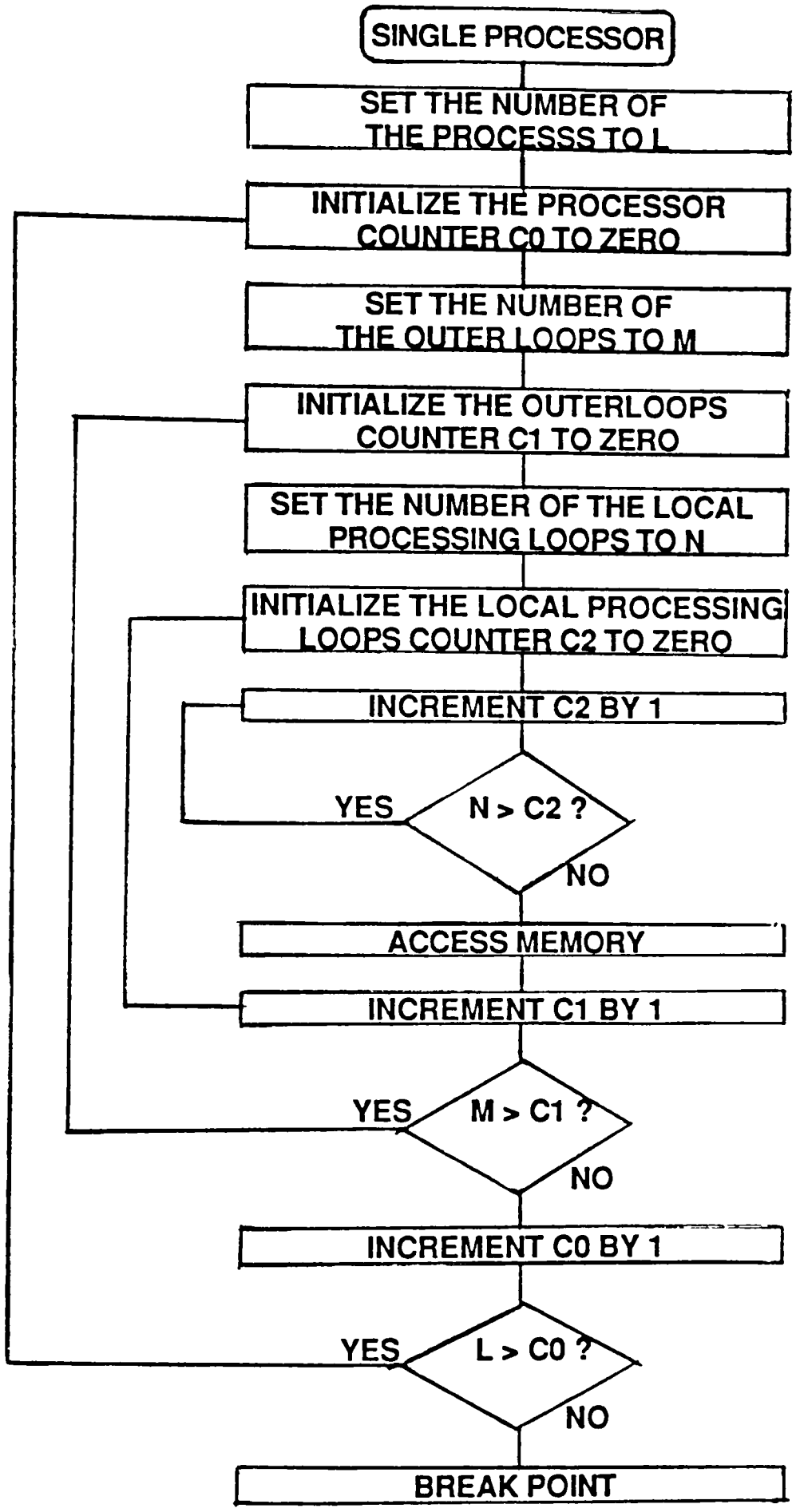

Figure 14: Flow Chart for a Single Processor

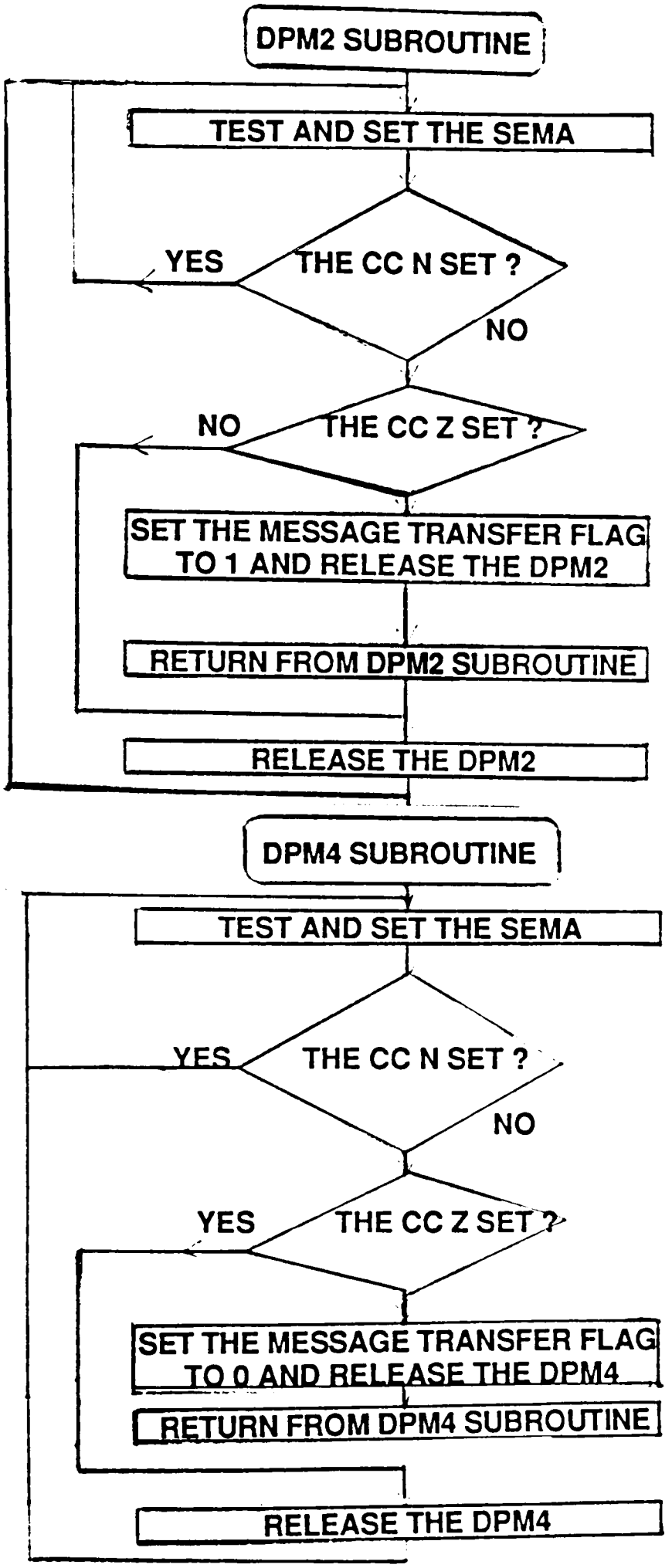

Figure 15: Flow Chart for the DPM Subroutines

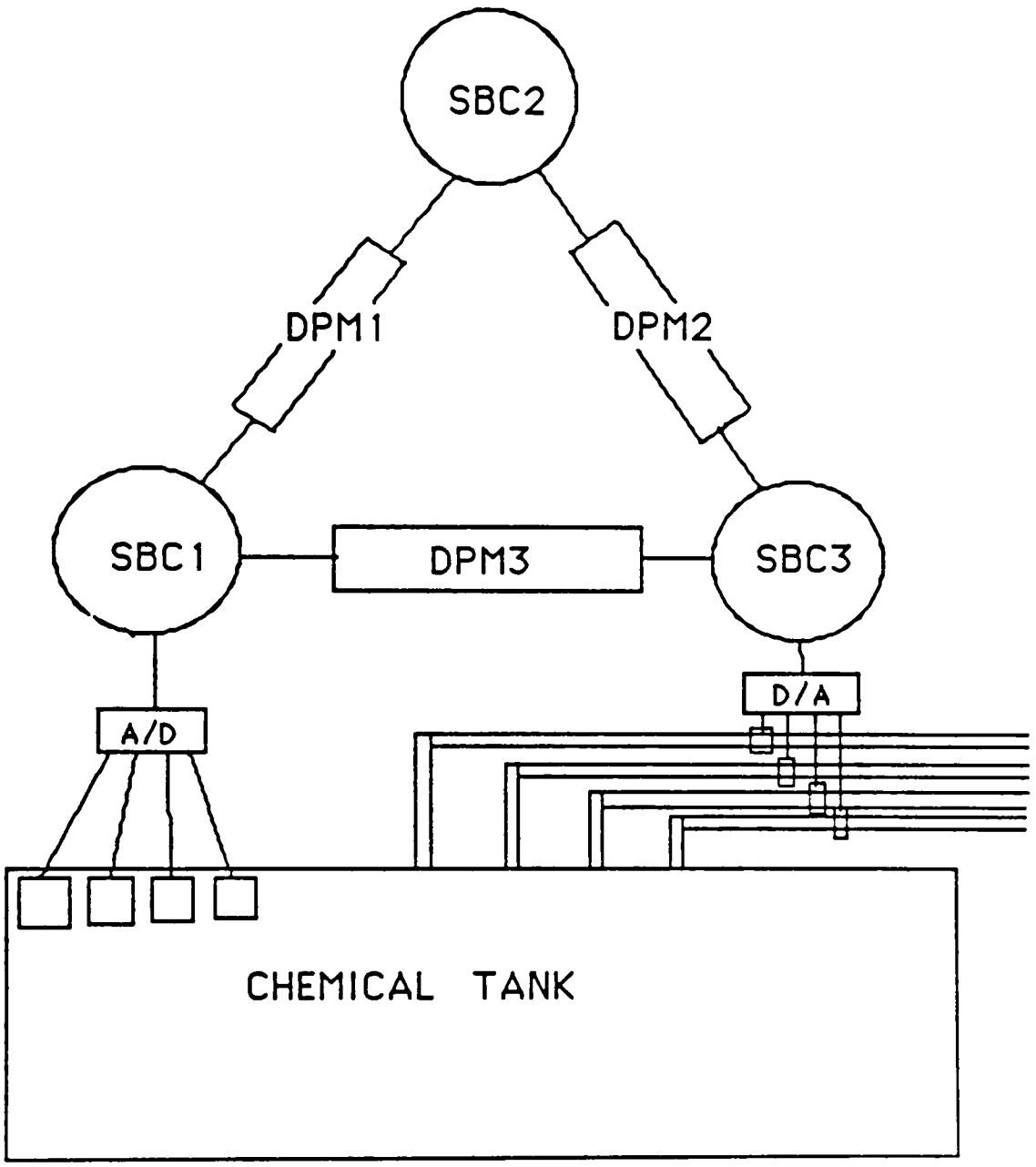

Figure 16: An Application of the System Designed

SBC3 takes messages from DPM2, calculates necessary changes in the process parameters, and sends control signals to the digital-analog convertor, which has four sensors connected to it. These sensors control valves to adjust the process parameters to some values. The changes made by SBC3 are also sent to SBC1 through DPM3. Thus, SBC1 may use these changes to compare with current values input from its sensors to see if the changes have been reflected in the tank.

Sara Lee Bakeries, Deerfield, Illinois has a multiple processor system with a common memory that performs a similar process control in its production procedures [WEIN86]. The multiple microprocessor system with DPMs, developed in this research, if used in a plant like Sara Lee Bakeries, could speed up these control processes by eliminating bus contention and reducing memory conflicts. This would produce more effective data acquisition and control signals, which should result in higher quality control and better products.

5.3 The Test Results

At the termination of each test, each SBC reached a breakpoint. All register contents, including the final values of M and N, were displayed on the ACT-5A screen. In every case, the system ran correctly, producing the correct results. In order to compare the increased throughput of

our system, we embedded an almost identical program in a main loop which executed it three times. This enlarged program was then run in a single SBC system. For all tests we collected and compared elapsed execution times for various numbers of inner loops. The results are shown in Tables 2 through 5. The ratios of the elapsed times for the two systems are shown in Tables 6 and 7. All tabulated data is displayed graphically in Figures 17 and 18.

The elapsed times were measured with a hand-held stopwatch. Five runs were made for each value of N, the number of inner loops. The averages of elapsed times as a function of N are plotted in Figures 17 and 18. As the figures clearly show, the elapsed times are nearly perfect linear functions of N. We calculated, tabulated and plotted the ratios of elapsed times for our system and for the single SBC system. The ratios improved as N increased and seem to be asymptotic to the value 3, which is the theoretical maximum increase in throughput for a multiple processor system of three SBCs.

Execution times of the test programs with ten thousand outer loops and only one local processing loop were then measured as a function of lock loops, which is the number of times an SBC will try to access a dual port memory before it gives up. Then the semaphore bit in the DPM is set to zero

TABLE 2

Execution Times of Three SBCs in the Delta Configuration--Five Outer Loops

M	N	a1	a2	a3	a4	a5	AVG
5	10000	0.87	0.89	0.83	0.80	0.88	0.85
5	20000	1.37	1.30	1.34	1.31	1.31	1.33
5	30000	1.78	1.87	1.87	1.84	1.83	1.84
5	40000	2.40	2.36	2.34	2.34	2.34	2.36
5	50000	2.94	2.94	2.89	2.91	2.95	2.93
5	60000	3.45	3.48	3.46	3.48	3.49	3.47
5	70000	4.02	4.05	4.07	4.12	3.97	4.05
5	80000	4.56	4.62	4.52	4.52	4.58	4.56
5	90000	5.14	5.17	5.06	5.03	5.22	5.12
5	100000	5.56	5.58	5.65	5.61	5.72	5.62

Note: M--Number of outer loops.
N--Number of local processing loops.
a1, a2, a3, a4 and a5--Execution times in seconds.
AVG--Average of execution times.

TABLE 3

Execution Times of Three SBCs in the Delta Configuration--Ten Outer Loops

M	N	b1	b2	b3	b4	b5	AVG
10	10000	1.35	1.32	1.32	1.34	1.35	1.34
10	20000	2.41	2.36	2.42	2.41	2.38	2.40
10	30000	3.45	3.45	3.43	3.44	3.41	3.44
10	40000	4.57	4.49	4.51	4.47	4.49	4.51
10	50000	5.55	5.56	5.63	5.62	5.70	5.61
10	60000	6.65	6.66	6.80	6.85	6.86	6.76
10	70000	7.73	7.76	7.79	7.77	7.97	7.80
10	80000	8.85	8.77	8.77	8.73	8.75	8.77
10	90000	9.68	9.85	9.84	9.88	9.86	9.86
10	100000	10.89	10.89	10.97	10.89	10.92	10.91

Note: M--Number of outer loops.
N--Number of local processing loops.
b1, b2, b3, b4 and b5--Execution times in seconds.
AVG--Average of execution times.

TABLE 4

Execution Times of a Single Processor--Five Outer Loops and Three Main Loops

L	M	N	c1	c2	c3	c4	c5	AVG
3	5	10000	1.85	1.88	1.83	1.87	1.89	1.86
3	5	20000	3.45	3.56	3.46	3.44	4.48	3.68
3	5	30000	5.02	4.92	5.03	5.04	5.04	5.01
3	5	40000	6.60	6.72	6.65	6.70	6.78	6.71
3	5	50000	8.25	8.44	8.23	8.40	8.42	8.35
3	5	60000	9.85	9.84	9.80	9.86	9.86	9.84
3	5	70000	11.43	11.42	11.39	11.57	11.50	11.46
3	5	80000	13.00	12.98	12.98	12.98	13.02	12.99
3	5	90000	14.59	14.77	14.61	14.78	14.86	14.72
3	5	100000	16.22	16.18	16.16	16.20	16.24	16.20

Note: L--Number of main loops.
M--Number of outer loops.
N--Number of local processing loops.
c1, c2, c3, c4 and c5--Execution times in seconds
AVG--Average of execution times.

TABLE 5

Execution Times of a Single Processor--Ten Outer Loops and Three Main Loops

L	M	N	d1	d2	d3	d4	d5	AVG
3	10	10000	3.60	3.48	3.45	3.44	3.46	3.49
3	10	20000	6.63	6.66	6.69	6.64	6.60	6.64
3	10	30000	9.83	9.83	9.80	9.85	9.88	9.84
3	10	40000	13.02	12.99	13.04	13.06	13.08	13.04
3	10	50000	16.19	16.19	16.16	16.16	16.18	16.18
3	10	60000	19.38	19.30	19.35	19.36	19.36	19.35
3	10	70000	22.55	22.51	22.50	22.57	22.54	22.53
3	10	80000	25.69	25.72	25.69	25.74	25.70	25.71
3	10	90000	28.93	28.92	28.90	28.94	28.90	28.92
3	10	100000	32.10	32.11	32.06	32.13	32.14	32.11

Note: L--Number of Main Loops.
M--Number of outer loops.
N--Number of local processing loops.
d1, d2, d3, d4 and d5--Execution times in seconds.
AVG--Average of execution times.

TABLE 6

Ratios of Execution Times of a Single SBC with a Delta Configuration--Five Outer Loops

M	N	c1:a1	c2:a2	c3:a3	c4:a4	c5:a5	AVG
5	10000	2.13	2.08	2.20	2.20	2.15	2.15
5	20000	2.52	2.74	2.58	2.63	2.66	2.63
5	30000	2.82	2.63	2.69	2.74	2.75	2.73
5	40000	2.75	2.85	2.84	2.86	2.90	2.84
5	50000	2.81	2.87	2.85	2.89	2.85	2.85
5	60000	2.86	2.83	2.83	2.83	2.83	2.84
5	70000	2.84	2.82	2.80	2.81	2.90	2.83
5	80000	2.85	2.81	2.87	2.87	2.84	2.85
5	90000	2.84	2.86	2.89	2.94	2.85	2.88
5	100000	2.92	2.90	2.91	2.89	2.85	2.89

Note: M--Number of outer loops.
N--Number of local processing loops.
c1:a1, c2:a2, c3:a3, c4:a4 and c5:a5--Ratios of execution times.
AVG--Average of ratios.

TABLE 7

Ratios of Execution Times of a Single SBC with a Delta Configuration--Ten Outer Loops

M	N	d1/b1	d2/b2	d3/b3	d4/b4	d5/b5	AVG
10	10000	2.67	2.64	2.61	2.57	2.56	2.61
10	20000	2.75	2.82	2.76	2.76	2.77	2.77
10	30000	2.85	2.85	2.86	2.86	2.90	2.86
10	40000	2.85	2.89	2.89	2.92	2.91	2.89
10	50000	2.92	2.91	2.87	2.88	2.84	2.88
10	60000	2.91	2.90	2.86	2.83	2.82	2.86
10	70000	2.92	2.90	2.89	2.90	2.89	2.90
10	80000	2.90	2.93	2.93	2.95	2.94	2.93
10	90000	2.93	2.94	2.94	2.93	2.93	2.93
10	100000	2.95	2.95	2.92	2.95	2.94	2.94

Note: M--Number of outer loops.
N--Number of local processing loops.
d1:b1, d2:b2, d3:b3, d4:b4 and d5:b5--Ratios of execution times.
AVG--Average of ratios.

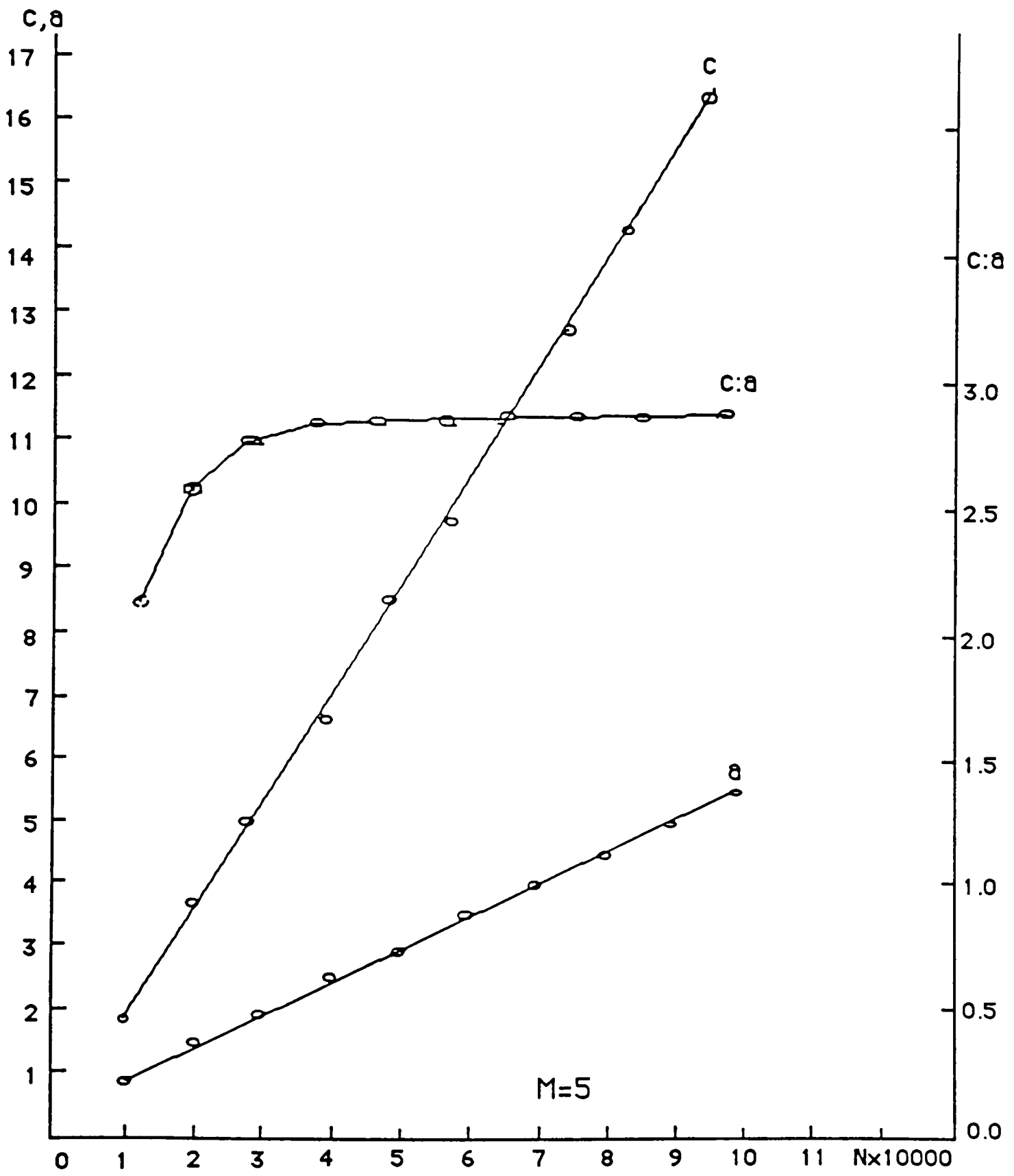

Figure 17: Execution Times, c and a, and Ratio c:a

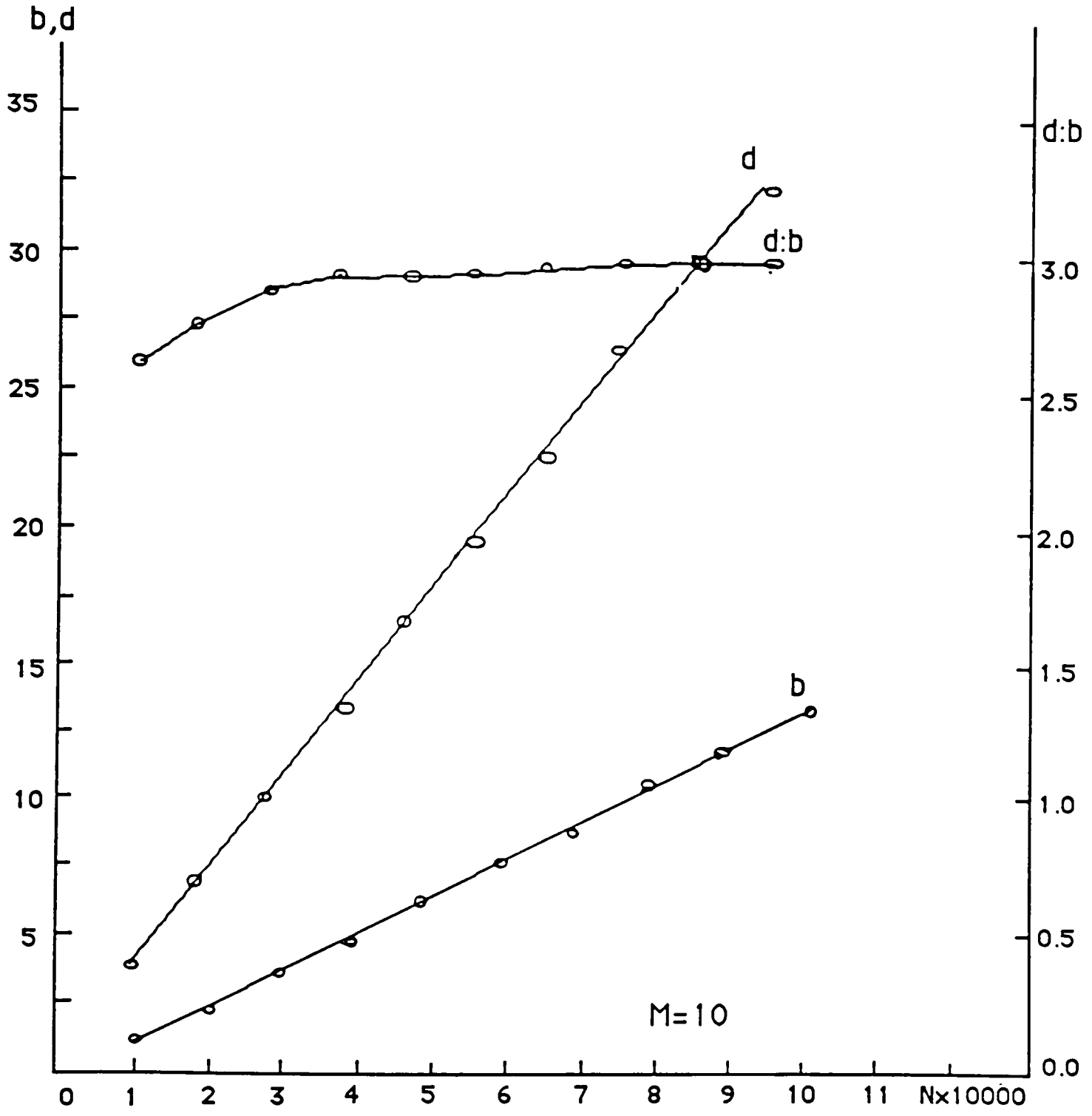

Figure 18: Execution Times, d and b, and Ratio d:b

and SBCs on both sides of the DPM start trying to access the DPM again. Overhead resulting from memory conflicts was estimated from these execution times. The data obtained from these measurement are listed in Table 8 and plotted in Figure 19. We analyze this data in the next chapter.

5.4 Preview

In the concluding chapter, Chapter 6, a summary of this research project is presented first. The results obtained from execution of the test programs are analyzed, and finally, future extensions of this research are discussed.

TABLE 8

Execution Times and Lock Loops

L	e1	e2	e3	e4	e5	AVG	MC
100	2.45	2.42	2.42	2.41	2.40	2.42	0.00
200	2.44	2.47	2.47	2.44	2.46	2.46	0.00
300	2.48	2.40	2.44	2.46	2.40	2.43	0.00
400	2.49	2.46	2.45	2.43	2.41	2.45	0.00
500	2.50	2.48	2.40	2.44	2.41	2.44	0.00
600	2.67	2.70	2.40	2.41	2.65	2.58	0.04
700	2.62	2.68	2.75	2.45	2.47	2.59	0.22
800	2.85	2.88	2.92	2.66	2.90	2.88	0.38
900	2.42	3.18	2.89	3.21	3.10	3.06	0.56
1000	2.88	3.24	2.84	3.81	3.25	3.12	0.76

Note: L--Number of lock loops.
e1, e2, e3, e4 and e5--Execution times in seconds
AVG--Average of execution times.
MC--The apparent increase caused by memory conflicts.
(Taken from Figure 18)
The lowest and highest values are not included in the average.

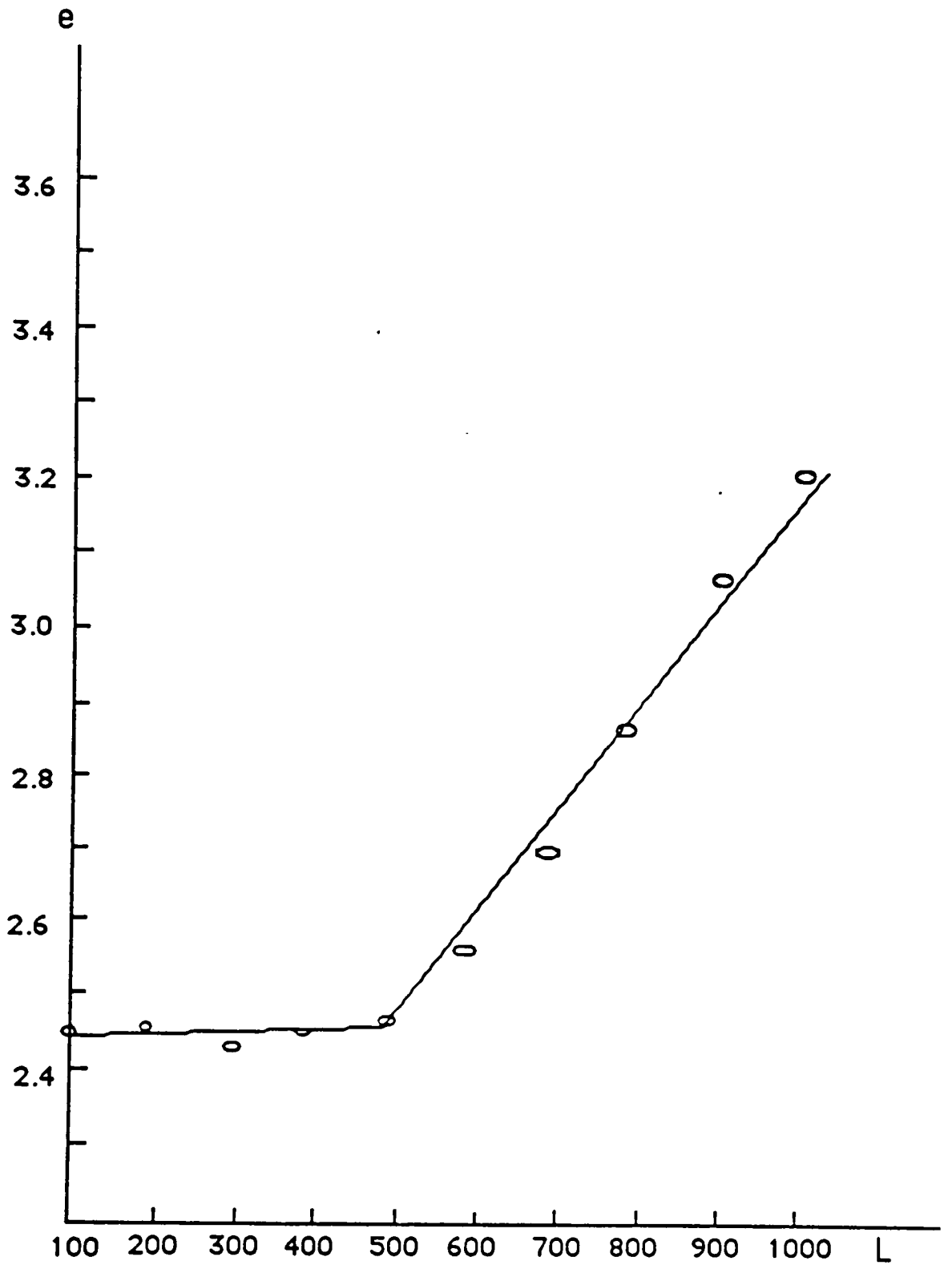

Figure 19: Execution Times e and Lock Loops L

CHAPTER VI

SYSTEM EVALUATION AND SUGGESTIONS FOR FUTURE RESEARCH

6.1 Summary

The major problems, which exist in many current, small scale, multiple microprocessor systems, using a common memory as the message exchange medium, are bus and memory conflicts. In this research, a multiple microprocessor system in a delta configuration has been designed and built. In this system, interprocessor communication is performed through dual port memories, which impose no bus conflicts and minimal memory conflicts on the system. There are three SBCs and three DPMs in the system. The MEX68KEC3, developed by the Motorola, Inc., is used as the functional unit, and the TMS9650, from Texas Instrument, Inc., is used as the dual port memory to transfer messages between SBCs.

A set of benchmark programs is used to measure the increase in throughput (which is the inverse of the execution time of a job performed by a computer system) and the amount of memory conflict which occurs in the system when extensive access is made to the DPMs. The test results are shown in Chapter 5 in Tables 2 through 8 and are plotted in

Figures 16 through 18. These results are used in the next section to evaluate the system.

6.2 Evaluation

As shown in Tables 6 and 7 and Figures 16 and 17, the increase in throughput ranges from 2.15 to 2.94. The lowest increase ratio, 2.15, results from the fact that the smaller number of local processing loops means a relatively higher DPM access frequency. This implies that there are more relative memory conflicts. Thus, the throughput ratio of the multiple processor system is lower when there is relatively less local processing compared with DPM accesses. For a large number of local processing loops, the increased throughput ratios are around 2.9 (See Table 7), which approach the theoretical maximum (3.0) for a multiple microprocessor system with three CPUs.

Memory conflicts are measured using 10000 outer loops and one local processing loop with different numbers of lock loops (Table 8 and Figure 18). When the number of lock loops increases from 100 to 500, the execution times are clustered around 2.44 seconds, which is the minimum execution time for executing the test program without lockout. When the number of lock loops increases to 1000, the execution time of the test program increases to about

3.20 seconds; 0.76 seconds seems to result from memory conflicts.

6.3 Future Research

A system with more SBCs and more DPMs may be developed to provide more processing power and more capability of reconfiguration. Software may be included in each local memory to implement dynamic reconfiguration.

We may find or build a different SBC which will allow more than the present seven DPMs to be connected to it.

The memory capacity of the TMS9650 chip used in this design is limited to two hundred and fifty six bytes. If capacity of memory in a DPM is increased to allow more messages to be transferred between two processors, the range of applications for this type of structure will be increased. Therefore, we need to find or build a different DPM which has a much greater memory capacity.

BIBLIOGRAPHY

[ALEX84] Alexandridis, Nikitas A., "Microprocessor System Design Concepts," 1984, Computer Science Press, Inc.

[BALP81] Baiph, Tom and John Black, "Multiprocessing Could Bring Out a System's Best, Application of VERSAbus and VERSAbus Products," 1981, Motorola Inc.

[BAUM84] Baumgartner, W. H., "Pulse Fundamentals and Small-Scale Digital Circuits," 1984, Reston Publishing Company, Inc., Pages 460-510.

[BORO79] Borovits, Israel, Seev Neuman, "Computer Systems Performance Evaluation," 1979, D.C. Heath and Company, Pages 9-26.

[BOZY82] Bozyigit, M. and Y. Paker, "A Topology Reconfiguration Mechanism for Distributed Computer Systems," 1982, The Computer Journal, Vol. 25, No. 1, Pages 87-92.

[CANN82] Cannon, Don L., "Fundamentals of Microcomputer Design," 1982, Texas Instruments Learning Center.

[DAVI81] Davis, Rex, "Prioritized Individually Vectored Interrupts for Multiple Peripheral Systems with MC68000," 1981, Motorola, Inc.

[ENSL74] Enslow, Philip H., Jr., editor, "Multiprocessors and Parallel Processing," 1974, John Wiley & Sons, Inc.

[FATH83] Fathi, Eli T. and Moshe Krieger, "Multiple Microprocessor Systems: What, Why, and When," 1983 March, IEEE, Pages 23-32.

[FERR78] Ferrari, Domenico, "Computer Systems Performance Evaluation," 1978, Prentice-Hall, Inc., Pages 26-89, 160-217.

[FLET80] Fletcher, William I., "An Engineering Approach to Digital Design," 1980, McGraw-Hill Company.

[GOSM82] Gosman, D., L. O. Hertzberger, G. Kieft, "The FAST Amsterdam Multiprocessor System Hardware," 1982 Feb., IEEE Transactions on Nuclear Science, Vol. NS-29, No. 1, Pages 314-318.

[GREB84] Grebene, Alan B., Micro-Linear Corporation, "Bipolar and MOS Analog Integrated Circuit Design," 1984, A Wiley- Interscience Publication, John Wiley & Sons, Pages 599-615.

[GRIN85] Grion, Jose M Llaberia, Mateo Valero Cortes, Enrique Herrada Lillo, and Jesus Labarta Mancho, "Analysis and Simulation of Multiplexed Single-bus Networks with and without Buffering," 1985 May, IEEE, Conference on Supercomputers, Pages 414-421.

[GROV82] Groves, Stan, "The Inter-Relationship between Access Time and Clock Rate in an MC68000 System," 1982, Motorola, Inc.

[HALL80] Hall, Douglas V., "Microprocessor and Digital Systems," 1980, McGraw-Hill, Inc.

[HEWL80] "HP64000 Logic Development System," Hewlett-Packard Company/Colorado Springs Division, 1980.

[HARM85] Harman, Thomas L. and Barbara Lawson, "The Motorola MC68000, Microprocessor Family, Assembly Language, Interface Design, and System Design," 1985, Prentice-Hall, Inc.

[HORD85] Hordoski, Michael F., "Design of Microprocessor Sensor & Control Systems," 1985, Reston Publishing Company, Inc.

[JOHN84] Johnson, James B. and Steve Kassel, "The Multibus Design Guidebook: Structures, Architectures, and Applications," 1984, McGraw - Hill, Inc.

[KART82] Kartashev, Svetlana P., ed., Steven I. Kartashev, "Designing and Programming Modern Computers and Systems," Vol. I, "LSI Modular Computer Systems," 1982, Prentice-Hall, Inc.

[KNIG78] Knight, W. W., and J. B. Williams, "A Dual Access Memory Architecture," 1978, General Electric Company.

[KOHA78] Kohavi, Zvi, "Switching and Finite Automata Theory," 1978, McGraw Book Company.

[LEIB85] Leibowitz, Burt H., John H. Carson, "Multiple Processor Systems for Real-Time Application," 1985, Prentice- Hall, Inc.

[MACG85] MacGregor, Doug, Motorola Inc., Jon Rubinstein, "A Performance Analysis of MC68020-based Systems," 1985 December, IEEE MICRO, Pages 50-71.

[MAPL85] Maples, Creve, "Analyzing Software Performance in a Multiprocessor Environment," 1985 July, IEEE Transactions on SOFTWARE, Pages 50-63.

[MARO82] Marovac, Nenad, "The Rotating Bus as a Basis for Interprocess Communication in Distributed Systems," 1982, The Computer Journal, Vol. 25, No. 1, Pages 22-31.

[MORR82] Morris, Michael F., and Paul F. Roth, "Computer Performance Evaluation, Tools and Techniques for Effective Analysis," 1982, Van Nostrand, Reinhold Company, Pages 74-133.

[MOTO81] "Motorola Microprocessors Data Manual," 1981, Motorola, Inc.

[MOTO82] "MC68000 Educational Computer Board User's Manual," 1982 January, Motorola, Inc.

[MOTO83A] "MC68000, 16-bit Microprocessor," 1983, Motorola, Inc.

[MOTO83B] "M68000 16/32-bit Microprocessor, Programmer's Reference Manual," 1983, fourth edition, Motorola, Inc.

[PATS81] Patstone, Walt, "16-bit-microprocessor Benchmarks--An Update with Explanations," 1981 September, ECN, Pages 169-182.

[RAO 82] Rao, Guthikonda V., "Microprocessor and Microcomputer Systems," 1982, Van Nostrand, Reinhold Company.

[RUHB81] Ruhberg, David L. and Michael C. Wood, "Multi-processor Controller Using the MC6809E and the MC68120," 1981, Microprocessor Application Engineering.

[RUSS77] Russo, Paul M., "Interprocessor Communication for Multi-microcomputer Systems," 1977 April, IEEE Transactions on Computers, Pages 67-76.

[SANG85] Sanguinetti, John, J. Kumar, "Performance of a Message- Based Multi-processor," 1985 May, IEEE Conference on Supercomputers, Pages 424-425.

[SATY80] Satyanarayanan, M., "Commercial Multiprocessing Systems," 1980 May, IEEE Computer, Pages 75-96.

[SIGN76] Signetics, "DATA MANUAL Logic, Memories, Interface, Analog, Microprocessor, Military," 1976, Signetics Corporation.

[SILV82] Silverman, Gordon, Avram Stundel, and John Lehman, "The Modular Multiprocessor - A Model for Laboratory Instrument Design," 1982 May, IEEE Micro, Pages 51-62.

[SIMT82] Smith, Kevin, Senior Editor, "System Harnesses up to Eight 68000s Achieving 4.8 MPS," 1982 Nov., Engineering Notes, Motorola, Inc.

[STAR80] Starnes, Thomas W., "Design Philosophy Behind Motorola's MC68000," Part 1 to Part 3, 1980, Motorola, Inc.

[STRI79] Stritter, Edward, Tom Gunter, "A Microprocessor Architecture for a Changing World: The Motorola 68000," 1979 Feb., IEEE Micro.

[TEDD84] Tedd, Mike, Stefano Crespi-Reghizzi, Antonio Natali, "Ada for Multi-Microprocessors," 1984, Cambridge University Press.

[TEXA84A] Texas Instruments, "High-speed CMOS Logic Data Books," 1984, Texas Instruments, Incorporated.

[TEXA84B] Texas Instruments Inc., "TMS9650 Multiprocessor Interface Data Manual," 1984, Texas Instruments Incorporated.

[TOUN81] Toung, Hoo-Min D. and Amar Gupta, "An Architectural Comparison of Contemporary 16-bit Microprocessors," 1981 May, IEEE Micro.

[VIVE82] Vivera, P., G. Conte, D. Del Corso, F. Gregoretti, and Pasero, "The Micro * Project: An Experience with a Multimicroprocessor System," 1982 May, IEEE Micro, Pages 38-50.

[WEIN86] Weiner, L. H., Private conversation, Texas Tech University, May, 1986.

APPENDIX A

THE HARDWARE DETAIL OF THE INTERFACE BOARD

APPENDIX A

THE HARDWARE DETAIL OF THE INTERFACE BOARD

The splitting and combining of the signals from the MC68000 to produce signals which are needed to drive the TMS9650 are analyzed in the following.

1. Activation of the Interface Board. The output signal E, E1 or E2, from the address decoder of the MC68000 is used to clear and activate the first Quad D-latch of the dual port interface board.
2. Generation of the CS* (Chip Select Low) Signal. The low output Q* of the first D-latch is used as the CS* signal to the TMS9650, and the high output Q is connected to the input of the second D-latch of this Quad D-latch.
3. Generation of the OE* (Output Enable Low) and WE* (Write Enable Low) Signals. The outputs of the second D-latch, Q* and Q, are combined with R/W* from MC68000, after one clock delay of 125 ns. The circuit is shown in Figure 20 (The Quad D-latchs are driven with 8 MHZ clock). these combined signals are used as two separate enable signals, OE* (Output Enable) and WE* (Write Enable), which are required by the TMS9650.

4. Selection of Two Paths. The output Q of the second D-latch is connected to two AND gates. After being combined with the ORed output and the NORed output of the address lines, A1 and A2, from the MC68000, it is used to select two signal paths.

(1) Generation of the DTACK* (Data Transfer Acknowledge Low) Signal. One path is selected when A1 or A2 or both are 1, which means that the MC68000 needs to access one of the TMS9650 registers other than the Data or Data/Inc register. In this case, the signal passes through third and fourth D-latch. The output of the fourth D-latch is used as the DTACK* to the MC68000, which is necessary for asynchronous I/O operation of the MC68000. The AND gates used to combine the DTACK*s to output to the MC68000 should have open collector outputs, since this signal is going to be ANDed with the other DTACK*s from the other peripheral devices in the system. That's why the IC chip SN74LS7409, instead of SN74LS7408, is used in the design.

(2) Activation of the second Quad D-latch. When both A1 and A2 are driven zero, the MC68000 needs to access the Data or Data/Inc register of TMS9650. The output Q of the second D-latch of the first Quad D-latch is

connected to the clear input of the second Quad D-latch to clear and activate it.

5. Generation of the DTACK* from the READY Signal. The output signal READY from the TMS9650 is connected to the input of the first D-latch of the second Quad D-latch. The four D-latches are cascaded, and the output Q* of the first D-latch is ORed with the output Q of the fourth D-latch to generate the DTACK* to the MC68000. The truth table for this generation is shown in Table 9 and the logic equation is as follows.

$$DTACK* = Q1* + Q4$$

6. The Bus Error Requirement. The READY is used to synchronize the TMS9650 and the MC68000 when a message transfer through RAM in TMS9650 is needed. If the RAM in the TMS9650 is accessed by the MC68000 from the remote side of the TMS9650, the MC68000 on the local side of the TMS9650 is put into a wait state through the READY signal. A bus cycle won't cause any Bus Error until a period of 40ms has elapsed. The period of 10 ms is about 40 clocks and is long enough for the MC68000 to perform regular memory access and release the dual port memory lock.

The diagram of the whole circuit is shown in Figure 21.

TABLE 9

Truth Table to Generate the DTACk*

READY Q1	Q1*	Q4	DTACK* Q1* + Q4
1	DC	DC	DC
1	DC	DC	DC
0	1	0	1
0	1	0	1
0	1	0	1
1	0	0	0
1	0	0	0
1	0	0	0
1	0	1	1

DC: Don't Care.

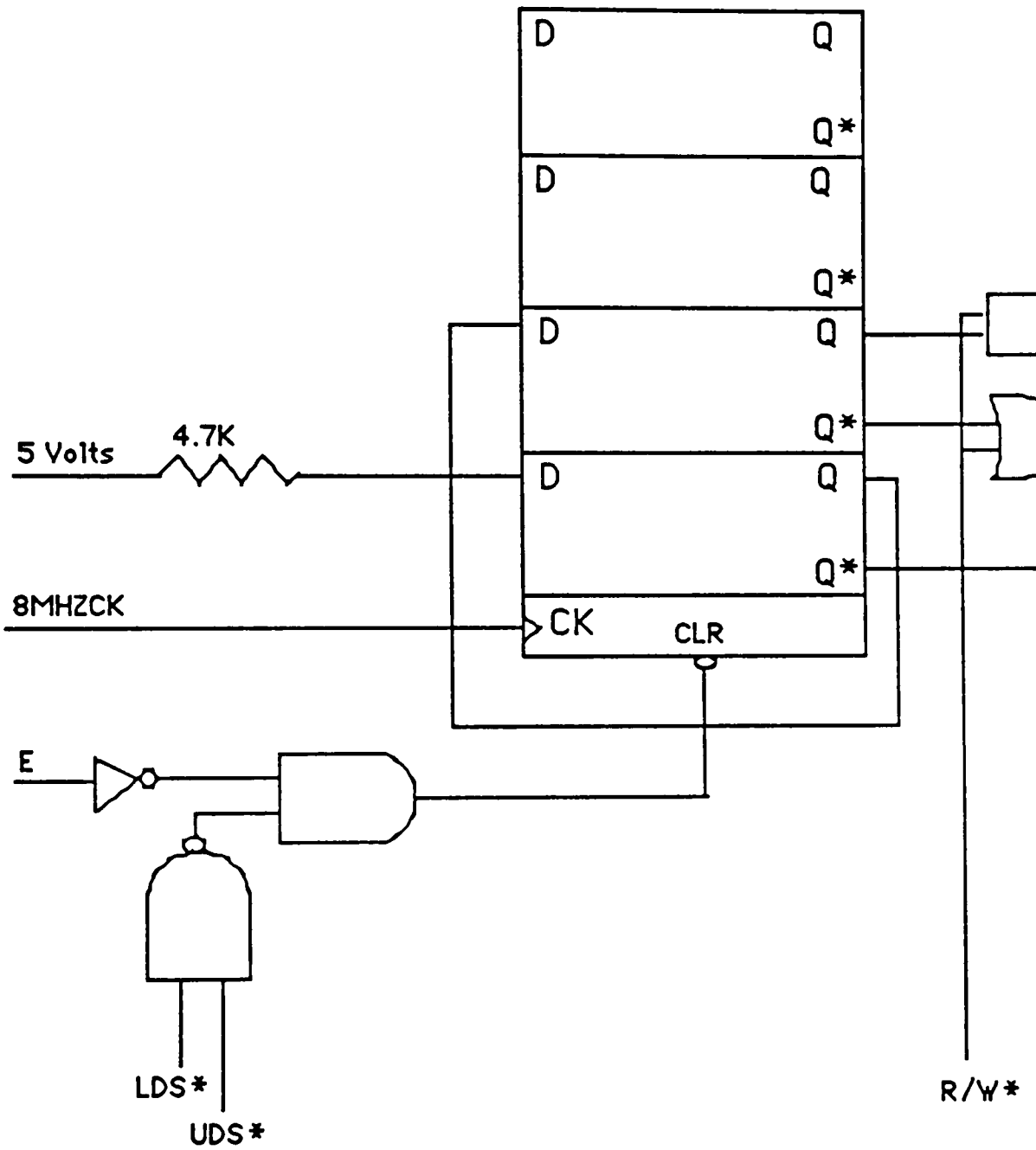

Figure 20: Generation of OE* and WE*

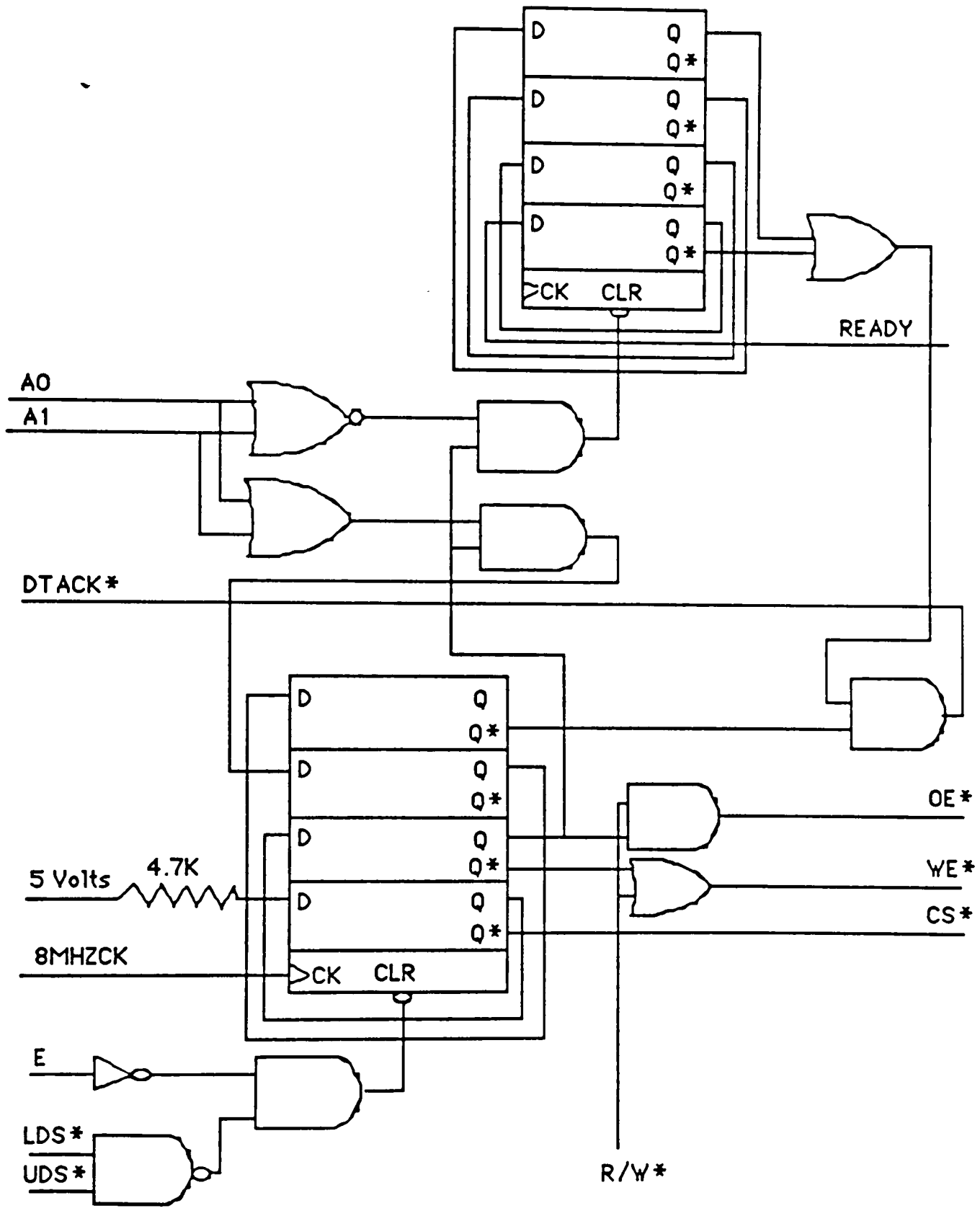

Figure 21: Diagram of the Interface Circuit

APPENDIX B

THE BENCHMARK PROGRAM

APPENDIX B

THE BENCHMARK PROGRAM

The following code is the benchmark to evaluate the system.

1. MULTIPLE PROCESSORS:

(a) PROCESSOR1:

```
            ADDRESS  ASSEMBLY LANGUAGE

            001000   MOVE.L   #10,D4
                     MOVE.L   #0,D5
                     MOVE.B   #2,$020008
                     MOVE.B   #1,$020009
            00101C   MOVE.L   #10000,D2
                     MOVE.L   #0,D3
            001028   ADD.L    #1,D3
                     CMP.L    D3,D2
                     BGT.L    $001028
                     BSR.S    $001064
                     BSR.S    $00109A
                     ADD.L    #1,D5
                     CMP.L    D5,D4
                     BGT.S    $00101C
                     MOVE.B   #0,$020008
                     MOVE.B   #1,$020009
```

```
                MOVE.B    #0,$040008
                MOVE.L    #0,$040009
001062          BRA.S     $001062

001064          MOVE.B    #0,$020008
                TAS.B     $020009
                BMI.S     $001064
                BNE.S     $001088
                MOVE.B    #0,$020008
                MOVE.B    #1,$020009
                RTS
001088          MOVE.B    #0,$020008
                MOVE.B    #1,$020009
                BRA.S     $001064

00109A          MOVE.B    #0,$040008
                TAS.B     $040009
                BMI.S     $00109A
                BNE.S     $0010BE
                MOVE.B    #0,$040008
                MOVE.B    #0,$040009
                BRA.S     $00109A
0010BE          MOVE.B    #0,$040008
                MOVE.B    #0,$040009
                RTS
```

(b) PROCESSOR2:

```
001000    MOVE.L    #0,D5
          MOVE.L    #10,D4
          MOVE.B    #0,$040008
          MOVE.B    #0,$040009
          MOVE.B    #2,$040008
          MOVE.B    #0,$040009
          MOVE.L    #0,D6
          MOVE.B    #2,$040008
          MOVE.B    #040009,D6
          CMP.B     #1,D6
          BNE.S     $001032
          MOVE.B    #2,$020008
          MOVE.B    #1,$020009
001056    MOVE.L    #10000,D2
          MOVE.L    #0,D3
001062    ADD.L     #1,D3
          CMP.L     D3,D2
          BGT.L     $001062
          BSR.S     $00109E
          BSR.S     #0010D4
          ADD.L     #1,D5
          CMP.L     D5,D4
          BGT.S     $001056
          MOVE.B    #0,$020008
```

```
                MOVE.B   #1,$020009
                MOVE.B   #0,$040008
                MOVE.B   #0,$040009
00109C          BRA.S    $00109C

00109E          MOVE.B   #0,$020008
                TAS.B    $020009
                BMI.S    $00109E
                BNE.S    $0010C2
                MOVE.B   #0,$020008
                MOVE.B   #1,$020009
                RTS
0010C2          MOVE.B   #0,$020008
                MOVE.B   #1,$020009
                BRA.S    $00109E

0010D4          MOVE.B   #0,$040008
                TAS.B    $040009
                BMI.S    $0010D4
                BNE.S    $0010F8
                MOVE.B   #0,$040008
                MOVE.B   #0,$040009
                BRA.S    $0010D4
0010F8          MOVE.B   #0,$040008
                MOVE.B   #0,$040009
                RTS
```

(c) PROCESSOR3:

```
          001000   MOVE.B   #0,$020008
                   MOVE.B   #0,$020009
                   MOVE.B   #0,$040008
                   MOVE.B   #0,$040009
                   MOVE.B   #2,$040008
                   MOVE.B   #0,$040009
                   MOVE.L   #0,D5
                   MOVE.L   #10,D4
                   MOVE.L   #0,D6
          001042   MOVE.B   #2,$040008
                   MOVE.B   $040009,D6
                   CMP.B    #1,D6
                   BNE.S    $001042
          001056   MOVE.L   #10000,D2
                   MOVE.L   #0,D3
          001062   ADD.L    #1,D3
                   CMP.L    D3,D2
                   BGT.L    $001062
                   BSR.S    $00109E
                   BSR.S    $0010D4
                   ADD.L    #1,D5
                   CMP.L    D5,D4
                   BGT.S    $001056
```

```
          MOVE.B    #0,$02000B
          MOVE.B    #1,$020009
          MOVE.B    #0,$04000B
          MOVE.B    #0,$040009
00109C    BRA.S     $00109C

00109E    MOVE.B    #0,$02000B
          TAS.B     $020009
          BMI.S     $00109E
          BNE.S     $0010C2
          MOVE.B    #0,$02000B
          MOVE.B    #1,$020009
          RTS

0010C2    MOVE.B    #0,$02000B
          MOVE.B    #1,$020009
          BRA.S     $00109E

0010D4    MOVE.B    #0,$04000B
          TAS.B     $040009
          BMI.S     $0010D4
          BNE.S     $0010F8
          MOVE.B    #0,$04000B
          MOVE.B    #0,$040009
          BRA.S     $0010D4
0010F8    MOVE.B    #0,$04000B
          MOVE.B    #0,$040009
```

PERMISSION TO COPY

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Texas Tech University, I agree that the Library and my major department shall make it freely available for research purposes. Permission to copy this thesis for scholarly purposes may be granted by the Director of the Library or my major professor. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my further written permission and that any user may be liable for copyright infringement.

Disagree (Permission not granted)	Agree (Permission granted)
_____	Chien-Chi Tzu
Student's signature	Student's signature
_____	June 25, 1986
Date	Date